UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

Receptor digital por software para sistemas globales de navegación

Adrian Miró Sanz

Tutor: Alberto Albiol Colomer

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2017-2018

Valencia, 1 de diciembre de 2018

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universitat Politècnica de València
Edificio 4D. Camino de Vera, s/n, 46022 Valencia
Tel. +34 96 387 71 90, ext. 77190
www.etsit.upv.es

Resumen

Los sistemas GPS son una de las ramas de las comunicaciones por satélite y pertenecen a un campo de estudio muy amplio y con un gran futuro. Es por ello que comprender el funcionamiento del sistema, desde la señal que transmiten hasta como son los componentes de los transmisores y receptores, es muy importante.

A partir de esta premisa nace el presente Trabajo Fin de Grado, en el que se espera entender más profundamente todo el sistema GPS hasta el punto de llegar a replicar y verificar uno de los bloques fundamentales de los receptores como es el de adquisición. La verificación del bloque diseñado se realizará a partir de una serie de simulaciones que permitan comprobar y mejorar el diseño realizado en MATLAB para, finalmente, aplicar el diseño a una señal GPS real previamente capturada con una antena y un SDR y utilizando el software GNSS-SDR.

Resum

Els sistemes GPS son una de les rames de les comunicacions per satèl·lit y pertanyen a un camp d'estudi molt extens y amb un gran futur. Es per aquesta raó que comprendre el funcionament del sistema, des de la senyal que es transmet fins a com son els components que formen els transmissors y receptors, es molt important.

A partir d'aquesta premissa neix aquest Projecte Fi de Grau, en el que es pretén entendre de manera més profunda tot el sistema GPS fins el punt de poder replicar y verificar un dels blocs fonamentals del receptors com es el d'adquisició. La verificació del bloc dissenyat es realitzarà a partir d'una sèrie de simulacions que permetran comprovar i millorar el disseny realitzat amb MATLAB per a, finalment, aplicar el disseny a una senyal GPS real prèviament capturada amb una antena i un SDR amb l'ús del software GNSS-SDR.

Abstract

GPS systems are one of the branches of satellite communications and belong to a very broad field of study with a great future. That is why understanding the operating of the system, form the signal they transmit to how the components of the transmitters and receivers are, is very important.

From this premise the present Final Degree Project is born, in which is expected to understand more deeply the GPS system to the point of replicate and verify one of the fundamental blocks of the receiver, such as the acquisition block. The verification of the designed block will be made from a series of simulations that will allow to check and improve the design made in MATLAB to, finally, apply the design to a real GPS signal previously captured with an antenna and an SDR using the GNSS-SDR software.

Índice general

Lista de abreviaturas

A
ADC: Analog-tp-Digital Converter
AWGN: Additive White Gaussian Noise

B
BPSK: Binary Phase Shift Keying

C
C/A: coarse acquisition code
CDMA: Code Division Multiple Access

D
DDC: Digital Down Converter
DFT: Direct Fourier Transform
DLL: Delay Locked Loop
DSSS: Direct Sequence Spread Spectrum
DSP: Digital Signal Processor

F
FEC: Forward Error Correction
FFT: Fast Fourier Transform

G
GNSS: Global Navigation Satellite System
GPS: Global Positioning System

H
HOW: HandOver Word

I
I: Phase IF: Intermediate Frequency

L
LHCP: Left Hand Circular Polarization
LOS: Line-Of-Sight
LFSR: Linear Feedback Shift Register

M
MCS: Monitoring Control Station
MEO: Medium Earth Orbit
MS: Monitoring Station

P

P(Y): Encrypted Precision Code
PCPS: Paralel Code Phase Search
PFSS: Parallel Frequency Space Search
PLL: Phase Locked Loop
PSP: Precise Positioning Service

Q

Q: Quadrature

R

RCHP: Right Hand Circular Polarization
RF: Radio Frequency

S

SDR: Software Defined Radio
SPS: Standard Positioning Service
SV: Satellite

T

TML: TeLeMetry
TOA: Time Of Arrival

U

UHF: Ultra High Frequency
UTC: Universal Time Clock

V

VSWR: Voltage Standing Wave Ratio

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

La radionavegación esta en auge y distintos países están apostando por desarrollar su propio sistema de posicionamiento por satélite, como es el caso del sistema **Galileo** para Europa, el **GLONASS** Ruso o el **Beidou** en China. Pero la localización de usuarios solamente es una de las múltiples aplicaciones de esta tecnología, que va desde, por ejemplo, aplicaciones militares hasta aplicaciones civiles de tipo lúdicas.

Al tratarse de un sistema en constante crecimiento y que todas las personas utilizan de forma cotidiana, no es de extrañar que los sistemas **GNSS** muevan grandes cantidades de dinero, solamente en Europa del 6 al 7 % del PIB depende del **GPS**.

Es por todo esto, que los sistemas **GNSS** son un buen tema sobre el que estudiar y sobre el que profundizar en su funcionamiento, tanto el del sistema como el de los receptores, pues es un campo interesante y con oportunidades para el futuro.

1.1 Motivación

Este trabajo Final de Grado nace por la idea de aplicar el uso de los **SDR**, vistos en la asignatura Tratamiento Digital de la Señal, a un sistema de comunicación por satélite, más concretamente el sistema **GPS**. Como se ha dicho anteriormente las tecnologías **GNSS** se utilizan de forma cotidiana en múltiples aplicaciones, es por ello que entender su funcionamiento es una gran oportunidad de cara al futuro.

Este sistema siempre ha despertado mi curiosidad acerca de su funcionamiento y de como se puede llegar a realiza el procesado de este tipo de señales que se han transmitido desde tanta distancia. Siendo el mundo de las comunicaciones por satélite amplio y complejo, este trabajo me ha dado la excusa de, no sólo poder iniciarme en él, sino el de mezclar una cosa que me gusta como son los **SDR** junto con un sistema que me entraña cierta curiosidad y que quiero conocer más profundamente.

1.2 Objetivos

El principal objetivo de este trabajo es entender tanto el funcionamiento del sistema **GPS** como el de los receptores que lo forman, con el fin de llegar implementar el bloque de adquisición de estos. El primer contacto para implementar este bloque de adquisición, será el de trabajar los distintos algoritmos con las señales **GPS** preparadas para esto y así validar y mejorar estos algoritmos. Finalmente, con los algoritmos puestos en práctica

se intentará obtener la adquisición de una señal **GPS** real capturada haciendo uso del programa GNSS-SDR y utilizando el **SDR** *HackRF*.

1.3 SDR

Los **SDR** o Software Defined Radio consituye un paradigma de diseño para las comunicaciones inalámbricas. Su creador, Joseph Mitola, definió el término a principios de los 90 como un tipo de comunicación por radio que podía ser reprogramado y reconfigurado a través de *software*. Mitola imaginó la idea de los **SDR** en los que sus únicos componentes *hardware*, en el caso del receptor, eran la antena y el convertidor analógico-digital o **ADC**.

La tecnología **SDR** o *Software Defined Radio* proporciona significativas ventajas con respecto a los sistemas tradicionales basados en *hardware*, pues componentes típicamente implementados en *hardware* en un sistema de radiocomunicaciones como son los mezcladores, filtros, moduladores/demoduladores, etc, son implementados en *software*.

Tradicionalmente, los sistemas radio estaban basados en *hardware* y sólo se podían modificar físicamente lo que suponi una flexibilidad de estos sistemas muy baja. En cambio, la tecnología **SDR** tiene componentes definidos por *software* lo que le confiere a esta tecnología más eficiencia, flexibilidad y mayor funcionalidad, pues sustituyendo o modificando sus programas *software* se consigue cambiar su funcionalidad. Esto permite especializar al **SDR** dependiendo de las necesidades de cada usuario.

Receptor SDR

En la figura 1.1 se observa el diagrama de bloques de un receptor **SDR**. El primer componentes que observamos, el *RF Tuner*, convierte la señal analógica a una frecuencia intermedia **IF**, esto equivaldria a los tres primeros bloques de un receptor superheterodino. A continuación, la señal analógica es digitalizada por el **ADC**. El bloque **DDC** o *Digital Down Converter* transforma las muestras a banda-base (**BB**) dispuestas apra que sena procesadas por un sistema digital (**DSP** o *DIgital Signal Porcessor*).

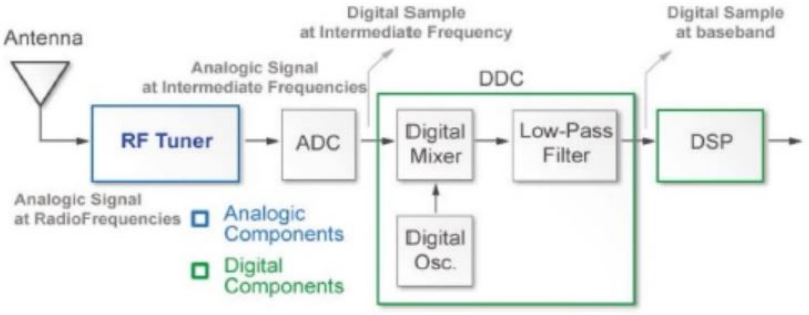

Figura 1.1: Diagrama de bloques de un receptor SDR

1.4 Metodología de trabajo

Para el desarrollo de este proyecto, ha sido imprescindible la preparación y planificación de un entorno de trabajo en el cual poder realizar las distintas pruebas.

1.4.1 Entorno de trabajo

La totalidad de este proyecto se ha realizado sobre el sistema operativo *Ubuntu*, versión 16.04 LTS, debido a que los programas utilizados para las capturas y validaciones de la señal **GPS** funcionan en este sistema operativo especifico.

GNSS-SDR

GNSS-SDR se trata de un programa que proporciona una interfaz para diferentes *front-end* e implementa cada etapa de los receptores **GPS** hasta la obtención de la posición del usuario. El diseño de este programa permite múltiples configuraciones, desde la posibilidad de trabajar con señales a tiempo real hasta la de trabajar con archivos de señales ya capturadas, permite la configuración de distintos algoritmos para el procesado de la señal, la configuración de los formatos de salida y ofrece interfaces para todas las señales intermedias. El programa esta, principalmente, desarrollado en C++ y utiliza GNU Radio para todas las funciones de procesado de señal de los receptores.

Figura 1.2: Logotipo de GNSS-SDR

GNSS-SDR se ejecuta en un ordenador personal y permite el uso de una variedad de *front-ends* disponibles comercialmente o personalizados, adaptando los distintos algoritmos a diferentes frecuencias de muestreo, frecuencias intermedias y resoluciones de las muestras. Esto hace posible la creación rápida y sencilla de receptores específicos distintos propósitos.

Como se trata de un programa orientado a objetos y con la idea de hacer los distintos bloques lo más independientes y genéricos posibles, GNSS-SDR se divide en los bloques que podemos ver en la figura 1.3, los cuales corresponden con los de un receptor **GPS**.

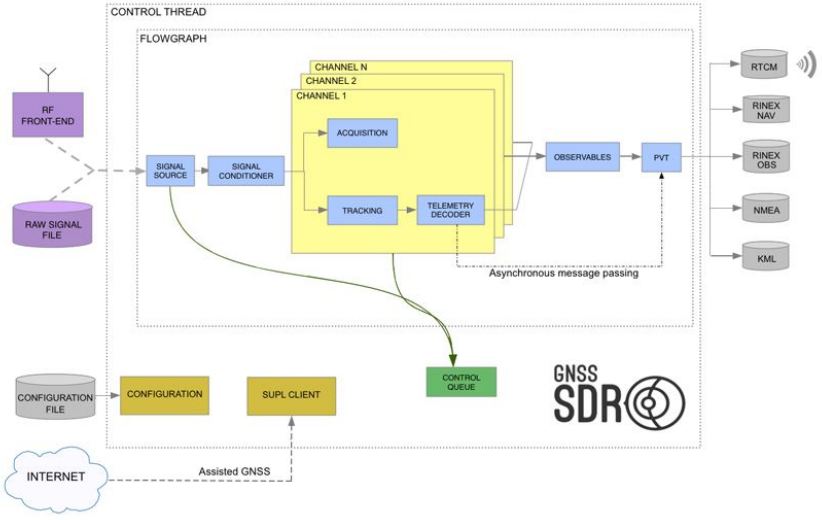

Figura 1.3: Esquema de los bloques que forman el programa GNSS-SDR

Matlab

Matlab (**Mat**rix **Lab**oratory) es un *software* de programación matemático, con un lenguaje propio, que realiza programación iterativa basada en matrices.

Con el fin de realizar mis propias simulaciones para el bloque de adquisición de un receptor **GPS**, se ha optado por utilizar Matlab debido a la familiaridad con este entorno de trabajo y a la obtención de una función que podemos ver en el Anexo (*cacode.m*), que será de mucha utilidad a lo largo de las distintas pruebas y simulaciones.

Figura 1.4: Logotipo de Matlab

1.4.2 Planificación temporal

Debido a lo amplio del tema a tratar y, por lo tanto, a la gran cantidad de información que debo buscar y asimilar, se ha realizado una planificación temporal del proyecto a lo largo de los distintos meses con el fin de alcanzar los propósitos marcados.

Julio:

- Información sobre el sistema GPS. La primera semana se dedica a recopilar información acerca del sistema **GPS**. Para este trabajo es necesario conocer muy bien como funciona y como son las señales del sistema.
- Información sobre GNSS-SDR y Ubuntu. Estos dos días se dedican a indagar acerca del programa GNSS-SDR que se encontró durante la búsqueda de información sobre el sistema **GPS**, y obtener información acerca del sistema operativo *Ubuntu*, pues como ya se ha comentado, el proyecto se ha realizado íntegramente sobre este sistema operativo debido al programa GNSS-SDR.

Figura 1.5: Planificación julio

Agosto

Figura 1.6: Planificación agosto

- Instalación de *Ubuntu* y GNSS-SDR. Instalación en mi ordenador de una partición con el nuevo sistema operativo para poder instalar después el programa GNSS-SDR y poder realizar las primeras pruebas con él.
- Redacción de un primer borrador de la memoria. Durante estos días se realizó una primera versión de la memoria, solamente la parte teórica del sistema **GPS**, tratando de poner en común toda la información obtenida en el mes de

julio.

Septiembre:

- Información acerca de los receptores. Con la intención de empezar a programar los algoritmos de adquisición, se buscó información acerca del funcionamiento de los receptores **GPS** y información más concreta acerca del bloque de adquisición de estos.
- Primeras pruebas con el GNSS-SDR. Antes de intentar capturar la señal **GPS** con el uso del programa GNSS-SDR, durante estos días se realizaron varios tutoriales y pruebas con el fin de entender el funcionamiento del programa y poder empezar ya a probar distintas configuraciones para capturar la señal
- Diseño de los primeros algoritmos. Diseño y validación de las primeras versiones de los algoritmos de adquisición para la señal **GPS** preparada para tal propósito.

Figura 1.7: Planificación septiembre

Octubre:

Figura 1.8: Planificación octubre

- Captura de la señal GPS. En las primeras semanas de octubre, se empezó a realizar la captura de la señal **GPS** utilizando distintas configuraciones y verificando las señales capturadas mediante diferentes métodos. Durante estos días se tuvieron problemas con la captura con ciertos parámetros de las configuraciones que retrasaron el trabajo.
- Diseño de la señal GPS simulada y segundas simulaciones. Diseño de un programa en Matlab que simula una señal **GPS** y la validación del mismo con otro programa que realiza la adquisición de la señal creada.

Noviembre:

- Adquisición de la señal GPS real. Al principio de este mes, se realizó la adquisición de la señal capturada en el mes de octubre y se hicieron distintas pruebas con algunos parámetros del algoritmo de adquisición para esta señal.
- Redacción de la memoria. El resto del mes se dedicó única y exclusivamente a la redacción de la memoria del TFG.

Figura 1.9: Planificación noviembre

Capítulo 2

Sistema GPS

2.1 Arquitectura del Sistema GPS

Los sistemas **GNSS** (*Global Navigation Satellite System*), entre ellos el sistema **GPS** (*Global Positioning System*), presentan una arquitectura claramente dividida en tres partes o segmentos: el segmento espacial, el segmento de control y el segmento de usuario (Ver Fig. 2.1). No se podría entender un sistema de **GNSS** sin alguna de estas tres partes.

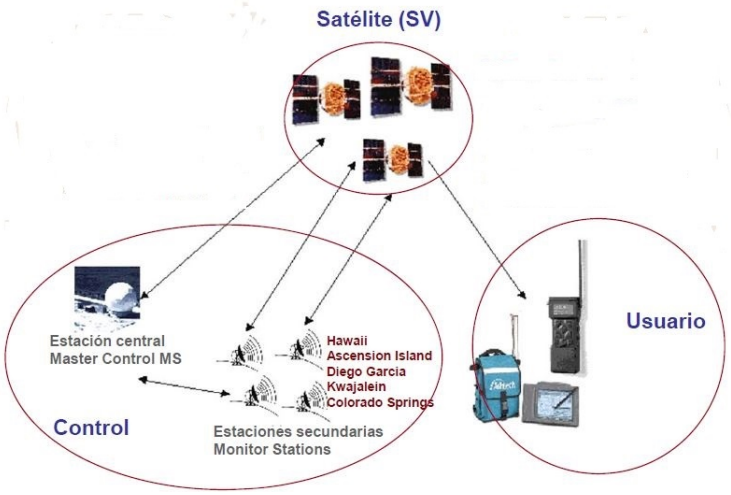

Figura 2.1: Arquitectura de un sistema GPS

2.1.1 Segmento espacial

El segmento espacial es el que se encuentra compuesto por los diferentes **SV** o satélites del sistema **GPS**. Un satélite no es más que un repetidor encargado de transmitir hacía los receptores en la superficie de la Tierra los datos de navegación, que incluyen, entre otras cosas, sus efemérides y diferentes correcciones de reloj que permiten al receptor estimar

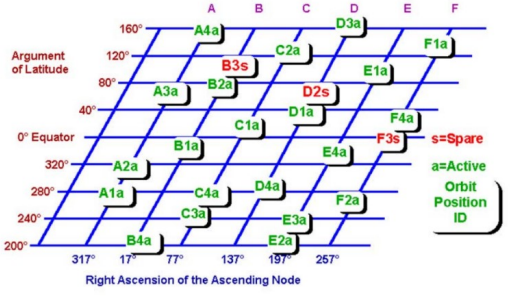

Figura 2.2: Reparto de satélites en las órbitas de GPS

la distancia que le separa del satélite y posteriormente calcular su propia posición.

Al principio la constelación del Sistema **GPS** constaba de 24 satélites nominales situados en 6 planos orbitales, con 4 satélites por plano, aunque actualmente dispone de una constelación de 32 satélites. Los satélites de la constelación siguen una órbita **MEO** (*Medium Earth Orbit*) casi circular con una altura de 26.650km y cuyo periodo es de, aproximadamente, 12 horas, proporcionando información de la ubicación y la hora en cualquier punto del planeta Tierra done haya **LOS** para cuatro o más satélites.

2.1.2 Segmento de control

El segmento de control es aquel formado por las distintas estaciones en tierra encargado del seguimiento y la monitorización de los distintos satélites del segmento espacial.

Sus funciones son las de garantizar el correcto funcionamiento del sistema, actualizando las efemérides de los satélites y aplicar distintas correcciones como en la posición orbital o en los relojes atómicos de los satélites.

El segmento de control se encuentra formado por:

- **Estaciones de Monitorización (MS)** que se encuentran repartidas por toda la superficie terrestre en emplazamientos cercanos al ecuador para maximizar su cobertura. Estas estaciones son las encargadas recoger información acerca del funcionamiento de los diferentes satélites del sistema.
- **Estación de Control Maestra (MCS)** es la encargada de centralizar toda la información recogida por las estaciones de monitorización y, además, lleva a cabo todos los procesos necesarios para mantener a los satélites en sus órbitas y realizar las correcciones de reloj pertinentes.

Figura 2.3: Posición de las MS y la MCS del sistema GPS

2.1.3 Segmento de usuario

El segmento de usuario es el que esta formado por todo el conjunto de receptores **GPS**, ya sean civiles o militares. Como se ha comentado anteriormente, estos son los encargados de calcular la distancia que le separa de cada satélite para así poder calcular su propia posición.

2.2 Funcionamiento

Con el fin de obtener la posición de un usuario sobre la superficie de la Tierra el sistema **GPS** se basa en el uso de un concepto llamado **TOA** (*Time Of Arrival*) o tiempo de llegada, esto lo que permite es obtener las distancias que separan al satélite que emite la señal y al usuario, con esta distancia y aplicando trilateralidad a cuatro o más satélites el receptor puede calcular su posición.

El **TOA** consiste en medir el tiempo que tarda en llegar a un receptor una señal emitida por un transmisor del que conocemos su posición. Este tiempo, referido al tiempo que la señal tarda en propagarse, es multiplicado por la velocidad de la luz para obtener la distancia entre ambos y junto a esta distancia a cuatro o más satélites se obtiene la posición del receptor.

Para el cálculo de la posición de un usuario mediante el concepto de trilateralidad, el cual se explicará de forma más profunda en el apartado *2.3*, es necesario que el receptor conozca de manera precisa la posición de todos los transmisores y, para esto, es necesario que los relojes de los transmisores y receptores estén perfectamente sincronizados. A partir de las medidas **TOA** se puede calcular el tiempo de propagación y, mediante este tiempo, se puede obtener una estimación de la posición del usuario con referencia al transmisor que envió la señal. Finalmente, asumiendo que la localización de los transmisores es conocida, el receptor puede obtener su posición haciendo uso de una serie de ecuaciones.

2.3 Cálculo de la posición de un usuario, trilateralidad

Como se ha comentado anteriormente el funcionamiento del sistema **GPS** se basa en el obtención de la distancia entre el satélite y el usuario a partir de los tiempos de propagación de las señales que estos emiten (**TOA**) y mediante el uso de un método matemático llamado trilateralidad el receptor puede calcular su posición.

El método matemático a partir del cual el receptor puede obtener su posición, usando la localización de los satélites y su distancia a ellos es la trilateración, la cual puede determinar posiciones relativas de objetos de forma análoga a lo que hace la tiangulación.

El método de trilateralidad necesita conocer la distancia entre el usuario y los, como mínimo, 4 o más satélites y para esto es necesario que exista una sincronización entre los relojes de los satélites y el del usuario.

2.3.1 Cálculo de la posición en 2D

Para poder explicar la trilateralidad de forma análoga a como se hace en el sistema **GPS**, vamos a utilizar un ejemplo en dos dimensiones.

Para este ejemplo, vamos a considerando el caso de un marinero en el mar que quiere conocer su posición relativa a una serie de sirenas que emiten un sonido, además, tendremos en cuenta una perfecta sincronización entre el marinero y las distintas sirenas utilizadas en el ejemplo.

El marinero empieza a contar el tiempo que tarda en escuchar el sonido de la sirena. El tiempo en el que tarda en escuchar dicho sonido es el tiempo que la señal tarda en propagarse, desde el momento en el que se produce hasta en el que lo escucha. Teniendo

en cuenta que la velocidad de propagación del sonido es de 335m/s si el marinero tarda 6s en escuchar el sonido, sabemos que se encuentra a 2010 metros de la primera sirena, a esta distancia la vamos a denotar como R1. Como solamente tenemos una medida de referencia, sabemos que el marinero se encuentra en algún lugar del círculo de radio R1 con centro la sirena 1. (Ver Fig. 2.4)

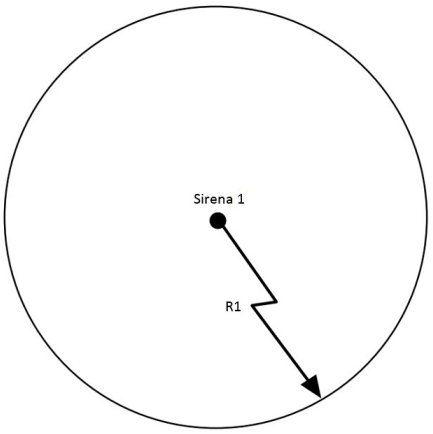

Figura 2.4: Determinación del rango por una sola fuente

Si suponemos que de forma simultánea el marinero escucha otra sirena y realiza lo mismo que en el caso anterior, este también se encuentra en el círculo de radio R2 y centro la sirena 2. Si juntamos las 2 posiciones relativas el marinero sólo puede encontrase en 2 puntos, los resultantes de la intersección de ambos círculos. (Ver Fig. 2.5)

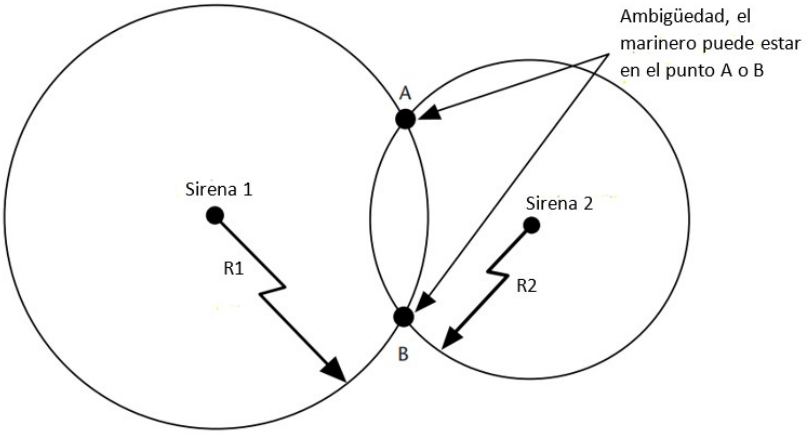

Figura 2.5: Ambigüedad resultante por la medida de dos fuentes

Para que el marinero pudiese acabar con esta ambigüedad y conocer su posición exacta sería necesario la utilización de una tercera sirena, esta sería la que soluciona la ambigüedad del caso anterior haciendo que el marinero se encuentra o en el punto A o en el B. (Ver Fig. 2.6)

Offset del reloj común y su compensación

En el ejemplo anterior hemos considerado que el reloj del marinero estaba perfectamente sincronizado con el tiempo base de las sirenas, aunque este no tiene por qué ser el caso. Si suponemos que el reloj del marinero se encuentra avanzado 1 segundo con respecto a las sirenas, significa que la distancia obtenida por el marinero será más grande debido a este segundo de *offset*. Los *offset* producidos son los mismos para todas

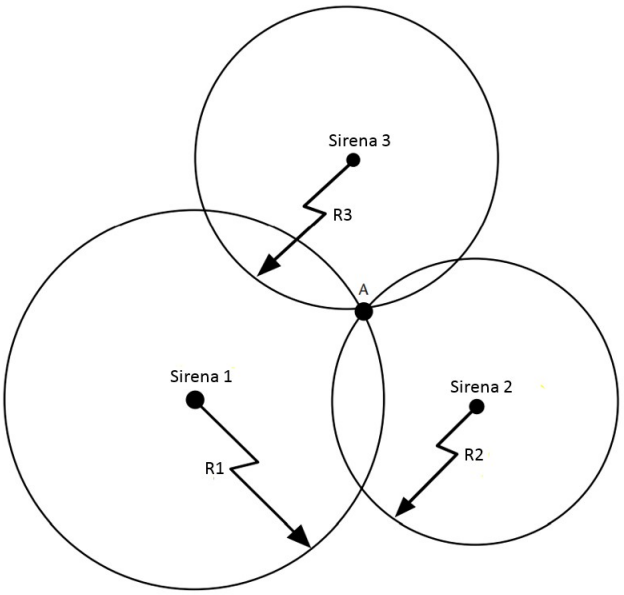

Figura 2.6: Ambigüedad resuelta añadiendo otra medida

las medidas debido a que este error se encuentra en el receptor y se denotará como ε en la figura 2.7. (Ver Fig. 2.7) La diferencia entre la intersección B, C y D y la verdadera posición A son funciones del *offset* del reloj en el receptor. Si este *offset* se pudiese quitar o compensar acabaríamos obteniendo el punto de intersección A.

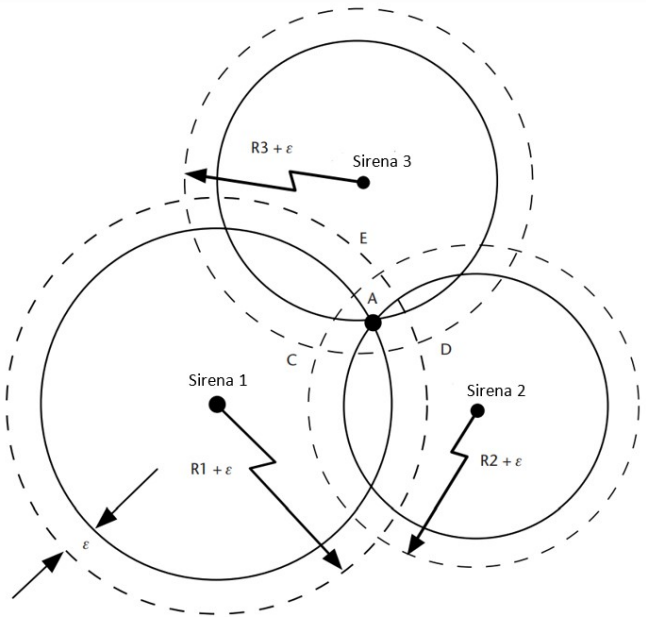

Figura 2.7: Efecto del *offset* del reloj del receptor en las medidas TOA

Con lo explicado anteriormente se puede plantear un sistema de ecuaciones en el que, al conocer las posiciones de las 3 sirenas, las únicas incógnitas serían la posición del usuario y el tiempo que tarda en propagarse cada una de las señales:

$$
\begin{aligned}
(x_1 - x)^2 + (y_1 - y)^2 &= (v\Delta t + (R_1 + \varepsilon))^2 \\
(x_2 - x)^2 + (y_2 - y)^2 &= (v\Delta t + (R_2 + \varepsilon))^2 \\
(x_3 - x)^2 + (y_3 - y)^2 &= (v\Delta t + (R_3 + \varepsilon))^2
\end{aligned}
\tag{2.1}
$$

Efecto de un error no común

Continuando con el mismo ejemplo, las medidas **TOA** realizadas puede que no sean perfectas debido a diferentes motivos, como pueden ser los debidos a errores producidos por la atmósfera, variaciones en el tiempo de *offset* de las sirenas cuando estas emiten el sonido, etc. Estos errores, a diferencia del que tenía el marinero con su reloj, no son comunes por lo que afectarán a las medidas de una forma única. La figura 2.8 (Ver Fig. 2.8) muestra el efecto que produce este error independiente para cada medida (ε_1 , ε_2 y ε_3). En lugar de tres círculos que intersecan en un punto, el marinero se encuentra en algún lugar del área de error delimitada.

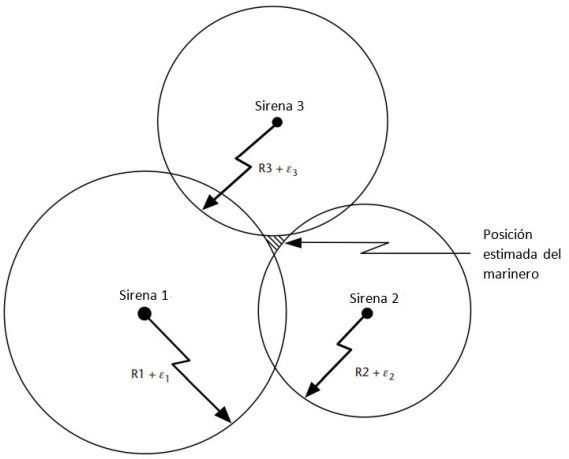

Figura 2.8: Efecto de un error independiente en cada fuente

2.3.2 Cálculo de la posición en 3D usando en el sistema GPS

El posicionado utilizado en el sistema **GPS** sigue el mismo concepto que el explicado anteriormente mediante el ejemplo del marinero pero en tres dimensiones. Sus principales diferencias son que en lugar de trabajar con círculos se trabaja con esferas, que las señales se propagan a la velocidad de la luz y de que la posición de cada satélite de la constelación se conoce con una gran precisión.

Como en el ejemplo del marinero, es necesario conocer las distancias que separan al receptor del satélite que emite la señal (su análogo en el ejemplo del marinero sería la sirena), pero en el caso del sistema **GPS** los satélites siguen una órbita a lo largo del tiempo razón por la cual la distancia entre estos y el receptor también varia. Es por esto que los satélites, junto a su posición, envían información de su reloj con el fin de que el receptor pueda estimar el tiempo de propagación, y al multiplicar este tiempo por la velocidad de la luz obtener la distancia que los separa. Cabría destacar que este calculo no sería una distancia al uso, sino una *pseudodistancia* pues los instantes de recepción y transmisión de la señal hacen referencia a escalas de tiempo distintas. Este error de sincronismo sera el que provoque la necesidad de usar un cuarto satélite para el cálculo de la posición al tener en cuenta la incertidumbre del tiempo.

Mediante esta *pseudodistancia*, la cual llamaremos R, se deduce que el receptor se encuentra en algún lugar de la superficie de la esfera de radio R, donde el satélite es el

centro de la esfera (Ver Fig. 2.9). Si de forma simultánea se hiciese lo mismo con otro satélite el receptor se encontraría o en el área producida al intersecar ambas esferas (Ver Fig. 2.9) o en un solo punto tangente a las dos. Este último caso solamente puede ocurrir si el receptor se encuentra colineal al satélite, aunque esto no se suele dar. El plano de intersección es perpendicular a la línea que une ambos satélites, tal como se observa en la figura 2.10. (Ver Fig. 2.10)

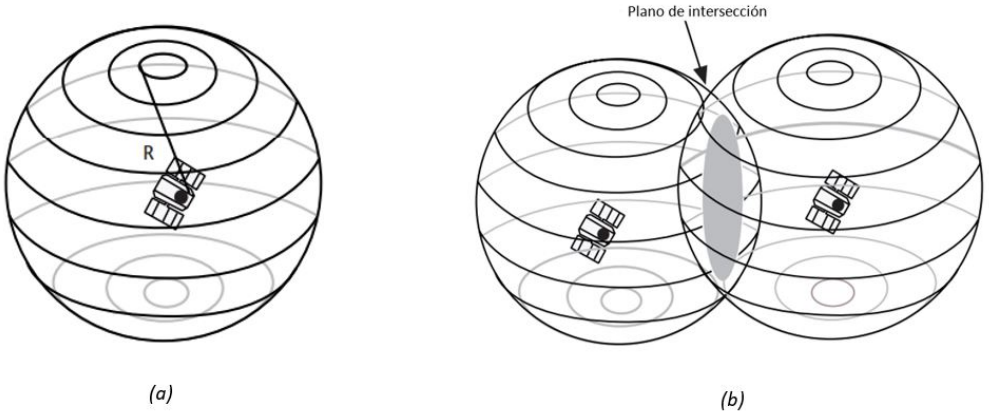

Figura 2.9: (a) Receptor situado en la superficie de la Tierra - (b) Receptor situado en la superficie sombreada, resultado de la intersección de las dos esferas

Repitiendo la medida usando un tercer satélite el receptor se encontrará en la intersección del perímetro del circulo con la superficie de la esfera. El resultado será el de dos puntos, aunque solamente uno será el correcto (Ver Fig. 2.10). Se puede observar que los puntos candidatos son imágenes el uno del otro respecto al plano de los satélites (Ver Fig. 2.10) Para un receptor en la superficie de la tierra uno de los dos puntos obtenidos se podrá descartar al ser absurdo.

Con sólo tres satélites se podría conocer la posición de un usuario en la superficie terrestre, ya que de los 2 puntos obtenidos uno se descarta debido a que es absurdo. A pesar de esto, para obtener el posicionamiento en el sistema **GPS** se necesitan 4 o más satélites, pues se debe tener en cuenta la incertidumbre temporal entre los satélites y el receptor.

Para el caso del posicionamiento 3D, con el uso de 4 satélites, tendiéramos el siguiente sistema de ecuaciones:

$$\begin{aligned}
(x_1 - x)^2 + (y_1 - y)^2 + (z_1 - z)^2 &= (c\Delta t + R_1)^2 \\
(x_2 - x)^2 + (y_2 - y)^2 + (z_2 - z)^2 &= (c\Delta t + R_2)^2 \\
(x_3 - x)^2 + (y_3 - y)^2 + (z_3 - z)^2 &= (c\Delta t + R_3)^2 \\
(x_4 - x)^2 + (y_4 - y)^2 + (z_4 - z)^2 &= (c\Delta t + R_4)^2
\end{aligned} \tag{2.2}$$

De todas las variables que se observan en el sistema anterior las coordenadas de los cuatro satélites (x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3) y (x_4, y_4, z_4) son conocidas debido a que el receptor recibe los datos de efemérides de cada satélite, y es propio del receptor calcular las distancias R_1, R_2, R_3 y R_4 entre él y el satélite en cuestión. Por lo tanto, tenemos un sistema con 4 ecuaciones y 4 incógnitas: las coordenadas del receptor (x, y, z) y el error de reloj del receptor.

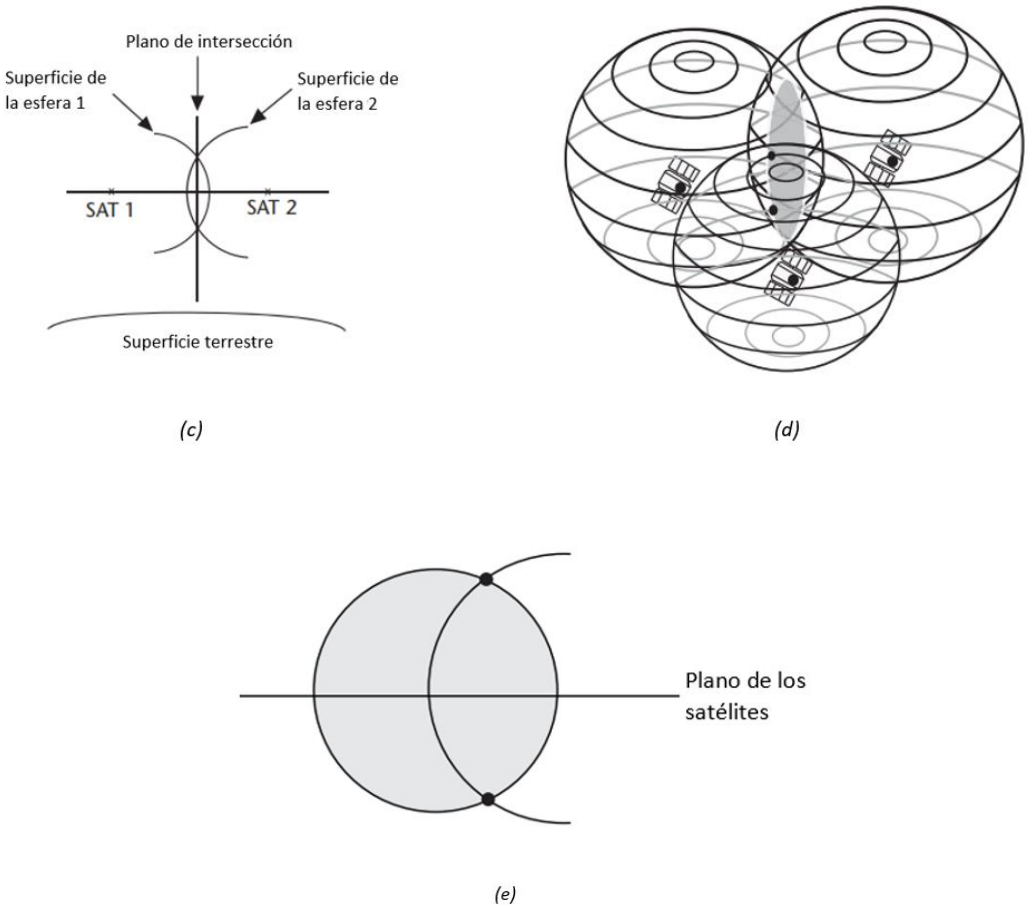

Figura 2.10: (c) Plano de intersección – (d) Receptor situado en uno de los dos puntos – (e) Receptor situado en uno de los dos puntos, normalmente uno será absurdo

2.4 Señales GPS

Desde que fuera plenamente operativo en 1993 el sistema **GPS** se ha ido perfeccionando y es algo que se usa de forma constante en el día a día de las personas. Este sistema nos proporciona dos tipos de servicio, referidos como **PSP** (*Precise Positioning Service*) y **SPS** (*Standard Positioning Service*). De estos dos servicios el **PSP** solamente se encuentra disponible para el uso militar mientras que el servicio **SPS** esta abierto a todos los usuarios civiles.

A lo largo del paso de los años las señales del **GPS** se han ido modernizando y ahora mismo este sistema consta de 3 *links* diferentes: L_1, L_2 y L_3. La portadora para cada *link* se obtiene a partir de una frecuencia fundamental $f_0 = 10{,}23MHz$. La Tabla 2.1 muestra los parámetros más significantes de las actuales y futuras señales **GPS**.

Vistas las diferentes señales que el sistema **GPS** posee, en este apartado nos vamos a centrar en conocer la estructura y como se forman de las señales **GPS** L1 C/A y L2 P. Para este cometido empezaremos conociendo sus componentes:

– Portadora: La señal **GPS** típicamente utilizan dos frecuencias portadoras en la banda de **UHF** (500MHz a 3GHz), estas dos portadoras llamadas L1 y L2 están formadas

Link	f_c [MHz]	PRN code	BW [MHz]	Pwr [dBW]	L_c [chips]	Code rate ($1/T_c$)	Mod.	Ch.	Servicio
L1	$f_0 \cdot 154$	C/A	20.46	-158.5	1023	$f_0/10$	BPSK(1)	single	SPS
	1575.42	P	20.46	-161.5	$6.1871 \cdot 10^{12}$	f_0	BPSK(10)	single	PPS
		L1CD	30.69	-158.25	10230	$f_0/10$	BOC(1,1)	data	Civil
		L1CP	30.69	-163	$10230 \cdot 1800$	$f_0/10$	BOC(1,1)	pilot	Civil
L2	$f_0 \cdot 120$	P	20.46	-158.5	$6.1871 \cdot 10^{12}$	f_0	BPSK(10)	single	PPS
	1227.60	L2CM	24	-158.5	10230	$f_0/10$	BPSK(1)	single	Civil
		L2CL	24	-158.5	$10230 \cdot 75$	$f_0/10$	BPSK(1)	pilot	Civil
L5	$f_c \cdot 115$	L5I	24	-157.9	$10230 \cdot 10$	f_0	BPSK(10)	data	SoL
	1176.45	L5Q	24	-157.9	$10230 \cdot 20$	f_0	BPSK(10)	pilot	SoL

Tabla 2.1: Lista señales GPS, códigos y modulación

a partir de una frecuencia fundamental $f_0 = 10{,}23MHz$.

$$f_{L_1} = 154 f_0 = 1575{,}42MHz \tag{2.3}$$

$$f_{L_2} = 120 f_0 = 1227{,}60MHz \tag{2.4}$$

– Datos de navegación: Los datos de navegación contienen información acerca del satélite, se emite a 50bps y la información se actualiza desde las estaciones de tierra en el segmento de control. La estructura de los datos de navegación se explicará más adelante en el apartado ***2.4.2***.

– Secuencia de ensanchado: Cada satélite tiene dos secuencias de ensanchado o códigos únicos. El primero es el código **C/A** (*coarse acquisition code*), que se trata de un código no protegido y el otro código es el **P(Y)** (*encrypted precision code*) y se trata de un código militar protegido.

El código **C/A** es una secuencia de 1023 chips (un chip se corresponde a un bit y se llama así para enfatizar que no contiene información alguna). Este código se repite cada ms y modula a una velocidad de $f_0/10$=1.023 MHz. En cambio, el código **P(Y)** es más largo ($\approx 2.35 \cdot 10^4$ chips) y modula a una velocidad de 10.23MHz. Este código se repite cada 7 días. El código **C/A** se encuentra modulado en la portadora L1 mientras que el código **P(Y)** se modula en ambas portadoras, L1 y L2. En la sección 2.4.3. se explica en más detalle la generación de los códigos C/A.

2.4.1 Portadora

La señal **GPS**, dependiendo de que aplicación se requiera, se transmite en una de las 3 portadoras del sistema, aunque típicamente se utilicen dos de ellas. Estas tres frecuencias portadoras L1, L2 y L3 se encuentran en la la banda L de la región **UHF** y las tres se forman a partir de la frecuencia fundamental f_0.

– **L1**: Frecuencia portadora $f_{L_1} = 154\ f_0 = 1575.42$ MHz, es la utilizada para transmitir los códigos C/A, P y M (código utilizado para formar una nueva señal militar) aunque en esta sección solamente nos centraremos en los códigos C/A y P.

– **L2**: Frecuencia portadora $f_{L_2} = 120\ f_0 = 1227.60$ MHz, es la utilizada para transmitir los códigos P, M y L2C (código utilizado para formar una señal de uso civil que mejora la formada por los códigos C/A).

– **L5**: Frecuencia portadora $f_{L_5} = 115\ f_0 = 1176.45$ MHz, es la utilizada para transmitir el código *Safety-of-Life Signal* (código utilizado para formar una señal orientada a ofrecer servicios de búsqueda y salvamiento).

2.4.2 Datos de navegación

En esta sección se va a describir la estructura y el contenido de los datos de navegación. Tal y como se ha comentado anteriormente, estos datos se transmiten a 50 bps y son imprescindibles, pues proveen al receptor con la información necesaria para obtener la posición de cada uno de los satélites visibles y el tiempo de transmisión de cada una de las señales de navegación, a parte, estos datos contienen información auxiliar que pueden ser usados por el receptor.

$$Tasa_{bit} = 50bps \tag{2.5}$$

Por lo que se transmite 1 bit cada 20 ms:

$$T_{bit} = \frac{1}{Tasa_{bit}} = \frac{1}{50bps} = 20ms \tag{2.6}$$

Cada uno de los bits de los datos de navegación tienen una duración de 20 ms con un valor binario de 1 y -1, este valor binario se explicará más profundamente en el apartado *2.4.4*. En la figura 2.11 se puede observar una simulación de 100 bits del mensaje de navegación.

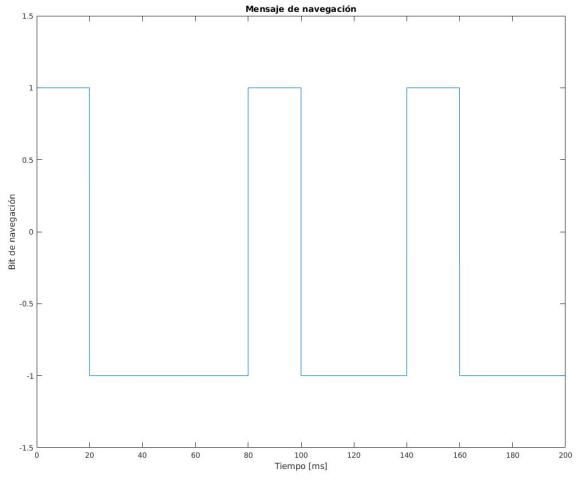

Figura 2.11: Mensaje de navegación

En cuanto a su estructura, los datos de navegación se transmiten en tramas siguiendo la estructura de la figura 2.12

Observando el esquema anterior se puede apreciar que para transmitir los datos de navegación completos, estos tardarán 12.5 minutos, o lo que es lo mismo 750 segundos, que atendiendo a la duración de cada bit de navegación tendremos un total de 37500 bits:

$$TotalBits_{DatosNavegación} = 750s\frac{1bit}{20ms} = 37500bits \tag{2.7}$$

Estos datos de navegación se distribuyen en un total de 25 tramas transmitidas durante los 750 segundos que tardan en transmitirse al completo los datos de navegación, esto supone que cada trama tarda 30 segundos en transmitir sus 1500 bits. Cada una de estas tramas contiene 5 subtramas de duración 6 segundos y 300 bits cada una. Finalmente, una subtrama contiene 10 palabras de duración 600 ms y de tamaño 30 bits. (Ver Tabla 2.2)

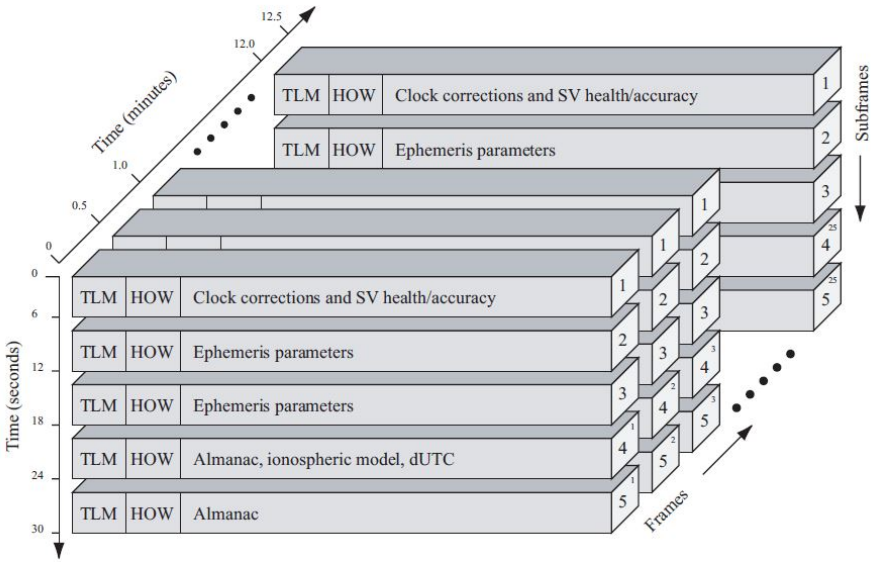

Figura 2.12: Estructura de los datos de navegación del sistema GPS

Tipo de dato	Cantidad en los datos de navegación	Duración unitaria	Bits unitarios
Trama	25	30 s	1500
Subtrama	$25 \cdot 5 = 125$	6 s	300
Palabra	$125 \cdot 10 = 1250$	600 ms	30

Tabla 2.2: Estructura de los datos de navegación

Como se ve en la figura 2.12 las dos primeras palabras de cada subtrama (los primeros 60 bits) son las palabras **TML** (TeLeMetry) y **HOW** (HandOver Word):

- TML: La palabra **TML** esta formada por 30 bits de los cuales los 8 primeros bits, los llamados preámbulo, son fijos (la secuencia 10001011 indica el inicio de la subtrama). Los 22 bits restantes se distribuyen de la siguiente forma: 14 bits para usuarios autorizados, 2 bits vacíos y los 6 restantes de paridad.
- HOW: La palabra **HOW** permite al receptor pasar de un código C/A a un código P.

A parte de las palabras **TML** y **HOW**, cada subtrama contiene 8 palabras de datos. En este caso, solamente vamos a describir los datos en las diferentes palabras y no la de todos los bits.

- Subtrama 1. La subtrama 1 contiene información acerca del reloj del satélite e información acerca de lo confiable o no de esta misma información. Esta información se actualiza en cada trama.
- Subtramas 2 y 3. Las subtramas 2 y 3 contiene las efemérides del satélite, compuestas por los parámetros orbitales y las anomalías. Esta información se actualiza en cada trama.
- Subtramas 4 y 5. Las subtramas 4 y 5 se repiten a los largo de cada trama, dando un total de 50 subtramas (esta es la razón del superínidice en el esquema de la figura 2.12). Estas dos subtramas contienen los datos del almanaque, que son datos del efemérides pero de menor precisión, de todos los satélites con el fin de facilitar al receptor la adquisición de cada uno de ellos, parámetros ionosféricos y parámetros

para relacionar el reloj del sistema **GPS** con el sistema **UTC** (*Universal Coordinated Time*).

2.4.3 Secuencia de ensanchado

Como se ha comentado en la introducción del apartado *2.4* cada satélite tiene 2 códigos únicos llamados secuencias de ensanchado, los cuales son una parte fundamental para la formación de la señal que estos envían. Estos códigos son: los **C/A**, modulados en la portadora L1; y los **P(Y)**, modulados en las portadoras L1 y L2. En las próximas secciones describiremos más profundamente los códigos **C/A**, su importancia en el sistema, su formación etc; pues en el trabajo nos hemos restringido únicamente a la señal civil.

Códigos C/A o Gold Codes

En esta sección vamos a describir las secuencias de ensanchado que se usan en la señal civil del sistema **GPS**. Esta secuencia de ensanchado pertenece a una familia de códigos llamada códigos Gold o *Gold codes*, pues fue Robert Gold quien las describió en 1967, aunque nos referiremos a ellos como *pseudo-random noise sequences* o, simplemente, **PRN** debido a sus características.

Gold sequence

Los códigos **PRN** transmitidos por los satélites del sistema **GPS** son secuencias pseudoaleatorias deterministas binarias cuyas propiedades son parecidas a las del ruido. Cada satélite genera un único código **C/A** debido a que, como todos los satélites del sistema transmiten en una misma portadora, estos códigos permiten que el receptor pueda identificar de que satélite es la señal recibida en la etapa de adquisición.

Los códigos **C/A** tienen una longitud de 1023 chips y constarán de 512 unos y 511 ceros pareciendo que se encuentran distribuidos de forma aleatoria, aunque toda la secuencia es generada de forma determinista. El código se repetirá de forma periódica cada 1 ms ($T_{CA} = 1ms$). En la jerga **GPS** a los bits que forman los códigos pseudoaleatorios se les denomina chips para enfatizar el hecho de que no contienen información. Por lo tanto, los **PRN** están formados por 1023 chips y tienen un periodo de 1 ms, con esta información se obtiene que el periodo de chip es de, aproximadamente, 1μs.

$$T_{chip} = \frac{T_{CA}}{1023chips} = \frac{1ms}{1023} = 0{,}977\mu s \approx 1\mu s \tag{2.8}$$

A partir de la inversa del T_{chip} podemos obtener la frecuencia de chip o *golda rate*.

$$f_{chip} = \frac{1}{T_{chip}} = \frac{1}{0{,}977\mu s} = 1{,}023MHz \tag{2.9}$$

En la figura (Ver Fig. 2.13) se observa la secuencia temporal de los 10 primeros chips del **PRN 10**, en esa figura se aprecia como cada chip que forma el código dura 1 μs.

Generación códigos Gold

La generación de los códigos Gold se puede observar en la figura 2.14, en ella se observa un diagrama del generador de códigos **C/A**. Este generador consta de dos **LFSR** (*Linear Feedback Shift Register*) llamados G_1 y G_2, estos **LFSR** pueden generar una secuencia de longitud máxima N = 2^n - 1 elementos, en los que la n es igual a 10 dando lugar a una longitud máxima de 1023 elementos. La secuencia generada es el resultado de la suma módulo-2 de una cierta combinación de los estados de los dos **LFSR**.

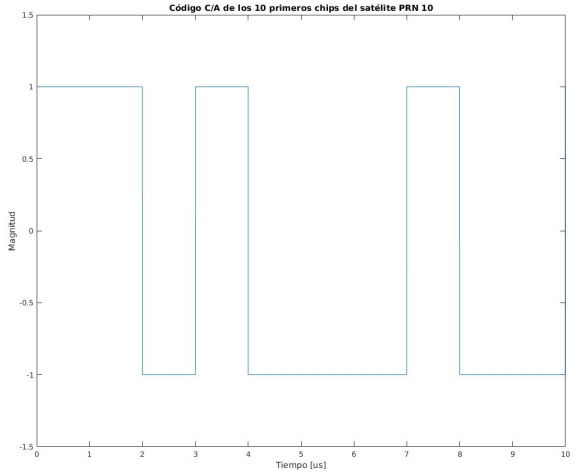

Figura 2.13: 10 primeros chips del PRN 10

En cada uno de los 1023 periodos, los dos **LFSR** son reseteados pasando todos sus estados a un valor de 1 haciendo que el código empieza de nuevo. La realimentación del primer **LFSR**, G_1, siempre tiene la configuración correspondiente al siguiente polinomio:

$$f(x) = 1 + x^3 + x^{10} \tag{2.10}$$

esto significa que el estado 3 y 10 son realimentados a la entrada del **LFSR**. De la misma forma el **LFSR** G_2 tiene el siguiente polinomio:

$$f(x) = 1 + x^2 + x^3 + x^6 + x^8 + x^9 + x^{10} \tag{2.11}$$

La generación de los distintos códigos **C/A** de cada uno de los satélites se realiza bit a bit como la suma módulo-2 de G_1 y G_2. Esta suma módulo-2 es la combinación de la propia salida de G_1 junto con una combinación de estados de G_2 únicos en cada satélite, esta selección de estados en G_2 se le llama *phase selection*. Por ejemplo, para crear el código **C/A** del **SV1** se combinarán la salida de G_1 con los estados 2 y 6 de G_2 (Ver Tabla 2.3). En esa tabla se observan las distintas combinaciones de estados en G_2 para cada satélite, al igual que los 10 primeros chips de cada combinación en octal.

ID del SV	Núm PRN	G2 - Phase Selection	Primeros 10 chips en octal	ID del SV	Núm PRN	G2 - Phase Selection	Primeros 10 chips en octal
1	1	$2 \oplus 6$	1440	17	17	$1 \oplus 4$	1156
2	2	$3 \oplus 7$	1620	18	18	$2 \oplus 5$	1467
3	3	$4 \oplus 8$	1710	19	19	$3 \oplus 6$	1633
4	4	$5 \oplus 9$	1744	20	20	$4 \oplus 7$	1715
5	5	$1 \oplus 9$	1133	21	21	$5 \oplus 8$	1746
6	6	$2 \oplus 10$	1455	22	22	$6 \oplus 9$	1763
7	7	$1 \oplus 8$	1131	23	23	$1 \oplus 3$	1063
8	8	$2 \oplus 9$	1454	24	24	$4 \oplus 6$	1706
9	9	$3 \oplus 10$	1626	25	25	$5 \oplus 7$	1743
10	10	$2 \oplus 3$	1504	26	26	$6 \oplus 8$	1761
11	11	$3 \oplus 4$	1642	27	27	$7 \oplus 9$	1770
12	12	$5 \oplus 6$	1750	28	28	$8 \oplus 10$	1774
13	13	$6 \oplus 7$	1764	29	29	$1 \oplus 6$	1127
14	14	$7 \oplus 8$	1772	30	30	$2 \oplus 7$	1453
15	15	$8 \oplus 9$	1775	31	31	$3 \oplus 8$	1625
16	16	$9 \oplus 10$	1776	32	32	$4 \oplus 9$	1712

Tabla 2.3: Asignación del Phase Selection de G2. La elección de diferentes estados para G2 genera distintos códigos C/A

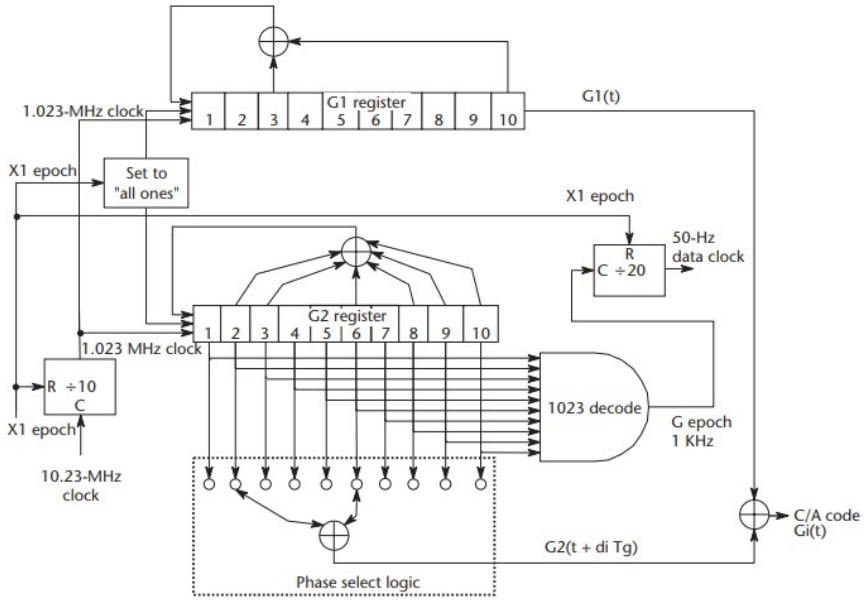

Figura 2.14: Generador del código C/A

Propiedades de la correlación

Los códigos Gold se escogen como secuencias de ensanchado para las señales **GPS** debido a sus características, aunque de entre todas ellas las más importantes son las relacionadas con su correlación. Estas propiedades con la correlación que tienen los códigos **C/A** se pueden definir como:

– Correlación cruzada prácticamente cero. Todos los códigos **C/A** se encuentran, prácticamente, incorrelados con el resto. Esto significa que, para dos códigos **C/A** C^i y C^k, siendo $i \neq k$, para los satélites i y k, su correlación cruzada es prácticamente cero, esto se puede expresar como:

$$r_{ik}(m) = \sum_{l=0}^{1022} C^i(l)C^k(l-m) \approx 0 \text{ para todo m} \tag{2.12}$$

En la figura 2.15 se aprecia como entre los códigos **C/A** de los satélites 1 y 2 su correlación es prácticamente nula.

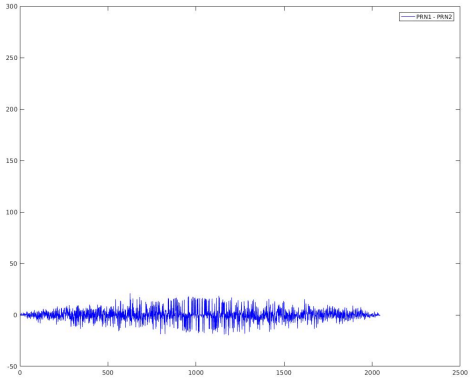

Figura 2.15: Correlación cruzada de los PRN 1 y PRN 2

– Correlación prácticamente cero excepto para un retardo nulo. Todos los códigos **C/A** se encuentran, prácticamente, incorrelados con ellos mismos excepto para un retardo nulo. Esto quiere decir que solamente cuando se multiplica la señal muestra a muestra por ella misma y están perfectamente alineadas, se produce un pico de correlación (Ver Fig. 2.16 (a)). La autocorrelación de un satélite k se puede expresar como:

$$r_{kk}(m) = \sum_{l=0}^{1022} C^k(l)C^k(l-m) \approx 0 \text{ para } |m| \geq 1 \tag{2.13}$$

En la figura 2.16 (b) se puede ver como, para el código **C/A** del satélite 1, el pico de autocorrelación disminuye si los códigos no se encuentran perfectamente alineados.

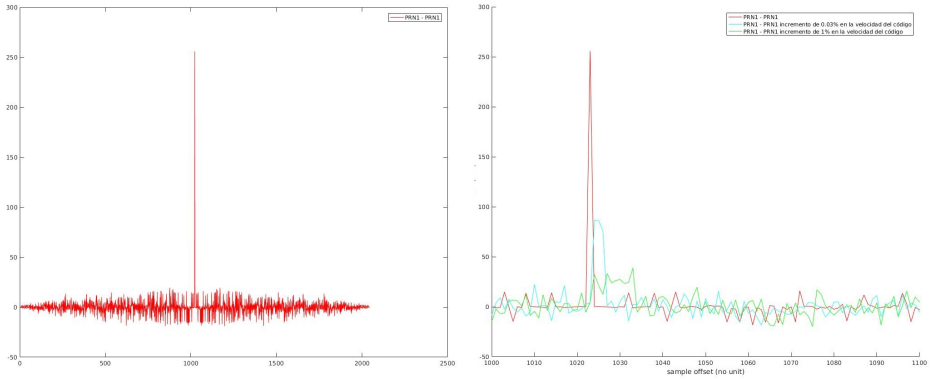

Figura 2.16: (a) Autocorrelación del PRN1 para un desfase cero - (b) Autocorrelación del PRN1 para un desfase distinto de cero

2.5 Estructura señal GPS

Una vez analizadas todas las partes que componen la señal **GPS**, en esta sección vamos a explicar como se combinan entre sí para formar la señal que los satélites transmiten, pero antes de explicar como se forma dicha señal, la figura 2.17 corresponde a una representación de la estructura de la señal **GPS** partiendo desde las 5 subtramas que forman una trama hasta los 1023 chips que forman 1 código **C/A**.

Viendo la figura 2.17 me gustaría hacer hincapié en que 1 bit del mensaje de navegación, el cual dura 20 ms, equivale a 20 códigos **C/A**, cuyo periodo es de 1 ms, es en esa transición en donde se combinan las tres partes de la señal **GPS** analizadas en los puntos *2.4.1*, *2.4.2* y *2.4.3* (Ver Fig. 2.18). Será en los siguientes apartados en donde se explique como se han combinado esas tres señales para formar los bits que se agrupan en palabras, las palabras en subtramas y las subtramas en tramas de la señal **GPS**.

2.5.1 Modulaciones empleadas

En el sistema **GPS** se utilizan dos modulaciones:

Una modulación digital llamada **BPSK** (*Binary Phase Shift Keying*), la cual produce un cambio de fase de 180^o en la portadora cuando, en los datos a modular, se produce un cambio de un 1 a un 0 o viceversa. La modulación **BPSK** la podemos ver en la figura 2.19 y se puede entender como la multiplicación de los datos con la señal de **RF** sin modular,

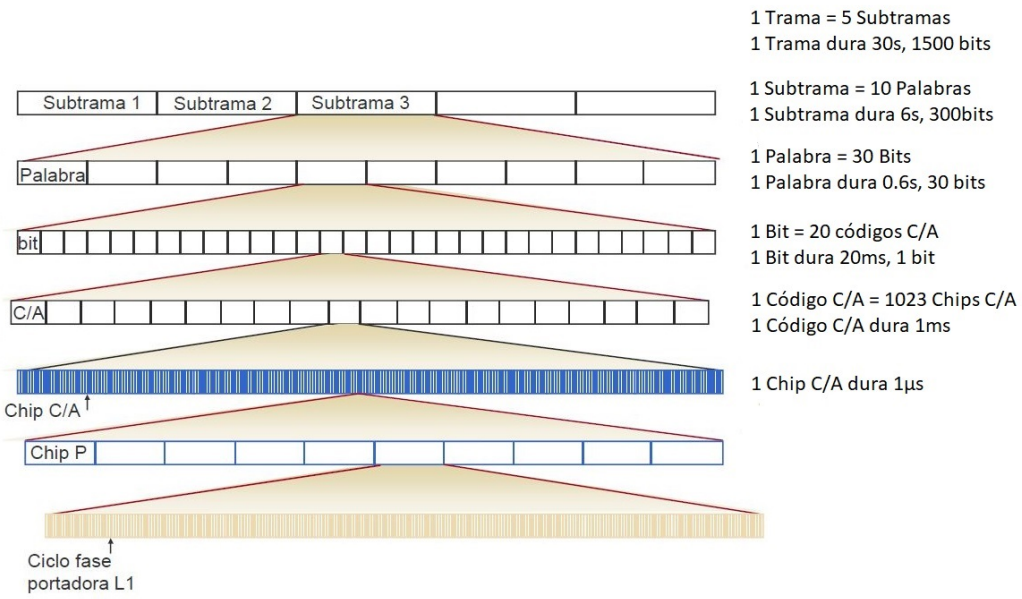

Figura 2.17: Estructura señal GPS

cuyos valores pueden tomar +1 o -1 para cada intervalo de $T_b = 1/R_b$ segundos, en donde R_b es la tasa de datos en bits por segundo.

Al igual que en otros muchos sistemas se emplea **FEC** (*Foward Error Correction*) como medida de corrección de errores en el receptor. Este método consiste en transmitir la señal junto a una serie de bits redundantes por el canal y mediante un cierto método permite al receptor corregir los errores producidos por el ruido, las interferencias o el *fading*, sin la necesidad de la retransmisión de los datos.

La otra modulación usada en el sistema, es la modulación **DSSS** o *Direct Secuence Spread Spectrum*, la cual es una extensión de la modulación **BPSK**. Tal como se observa en la figura 2.20 la modulación **DSSS** añade una tercera componente, la llamada secuencia de ensanchado o **PRN**, que hemos visto en el apartado *2.4.3*. Esta señal es similar al mensaje de navegación, que es la señal que transporta la información, pero con una velocidad de símbolo mucho mayor.

A la señal resultante de esta modulación se le llama de espectro ensanchado ya que el ancho de banda que la señal ocupa aumenta debido a la alta velocidad de símbolo del **PRN**. En general, el ancho de banda es proporcional al *chipping rate*.

Este tipo de modulación se realiza principalmente por tres razones:

– La primera, y más importante, es que las frecuentes inversiones de fase producidas por el **PRN** permiten al receptor realizar un posicionado más preciso.

  Para ilustrar esta ventaja hay que tener en cuenta que dos parámetros del mensaje de navegación que son inversamente proporcionales entre sí y que definen las transiciones que se producen en un cambio de bit:

$$B_{MensajeNavegación} = 50Hz \tag{2.14}$$

$$T_{bit} = 20ms \tag{2.15}$$

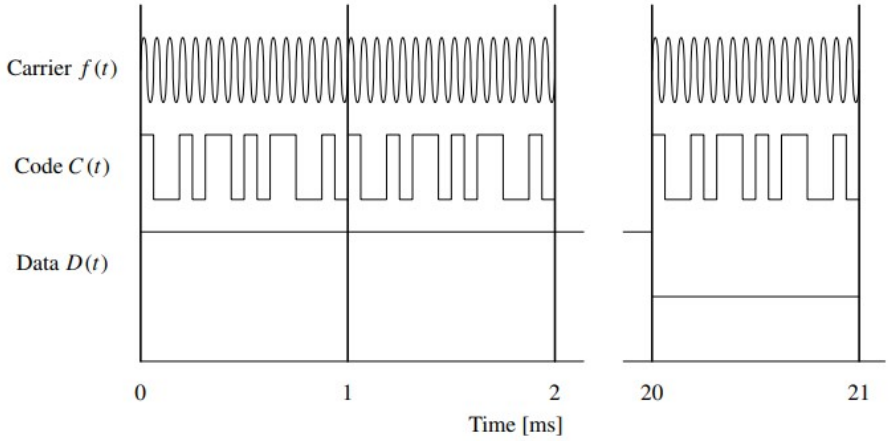

Figura 2.18: Estructura de la señal L1 - f(t) es la portadora, C(t) es la secuencia discreta del código C/A y que se repite cada ms y D(t) son los datos de navegación discretos en los que un bit dura 20ms

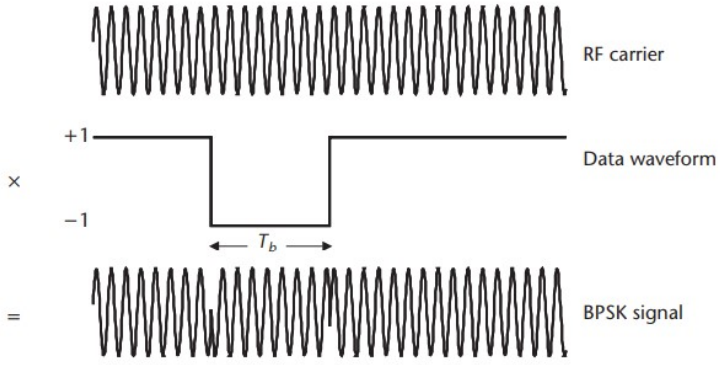

Figura 2.19: Modulación BPSK

Un sistema real no funciona mediante cambios abruptos como los observados en las figuras 2.11 y 2.13, por lo tanto, existen transiciones al pasar de un bit a otro. Por lo que resulta razonable pensar que el receptor cometerá menos errores en detección cuanto más abruptas sean estas transiciones y, por lo tanto, el sistema funcionará mejor.

La transiciones son inversamente proporcionales al ancho de banda, o lo que es lo mismo proporcionales al tiempo de bit, por lo que si deseamos unas transiciones pequeñas y abruptas debemos buscar unos valores grandes de ancho de banda. Comparado al $B_{CA} = 1$ MHz el valor del $B_{MensajeNavegación}$ es un valor pequeño, por lo que para conseguir estas transiciones abruptas trataremos de acercarnos lo máximo posible al valor de B_{CA}. Esto es lo que se consigue mediante la modulación **DSS**, transmitir el mensaje de navegación con un ancho de banda de 1 MHz.

– La segunda es que el uso de una serie de únicos y diferentes **PRN**, en un sistema bien diseñado, permite a los satélites transmitir de manera simultánea en la misma frecuencia portadora. Este hecho concuerda con una técnica de multiplexado denominada *Code Division Multiple Acces* o **CDMA** y es de gran importancia pues, como cada satélite tiene asignado un único **PRN**, el receptor puede identificar de que satélite es la señal recibida.

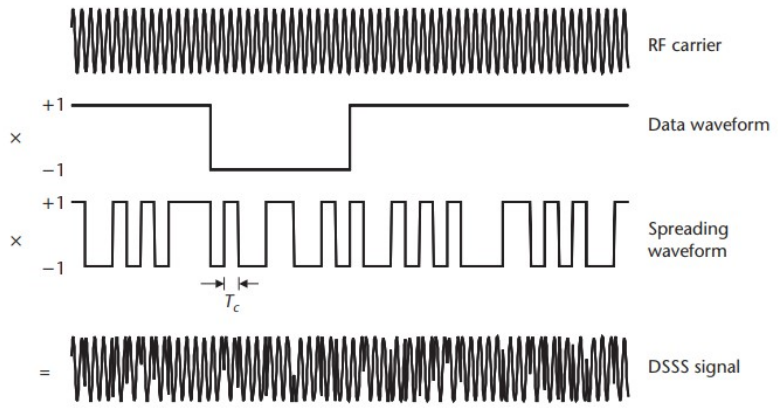

Figura 2.20: Modulación DSSS

– Finalmente, la tercera razón es que este tipo de modulación proporciona un buen rechazo a las interferencias entre bandas.

2.5.2 Formación de la señal

Con las diferentes partes que forman la señal **GPS**, la estructura de dicha señal y las modulaciones que emplea el sistema comentadas, solamente nos queda por ver el diagrama que explica como se aúna todo esto para formar la señal que los satélites envían, por lo que en esta sección vamos a explicar como se forma dicha señal. En la figura 2.21 tenemos un esquema que será nuestra guía para esta sección.

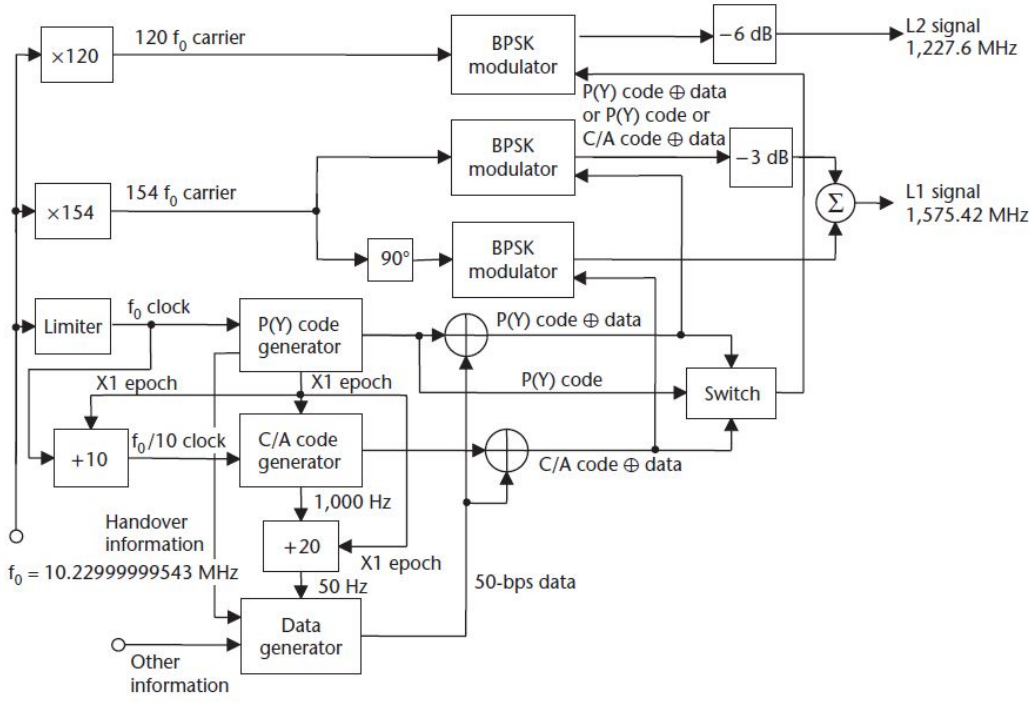

Figura 2.21: Generación de las señales en el satélite

Para empezar a explicar como se forma la señal vemos como el diagrama esta basado alrededor de la frecuencia maestra f_0 de 10.23 MHz, en realidad la frecuencia exacta es de 10.22999999543 MHz para ajustarla a los efectos relativistas y así conseguir que un

usuario en la Tierra tenga la frecuencia f_0 de 10.23 MHz. A partir de esta frecuencia maestra se forman las portadoras L1 y L2 multiplicar f_0 por 120 y 154 respectivamente. El limitador que podemos encontrar en el esquema se utiliza para estabilizar la frecuencia antes de usarla en los bloques que generarán los códigos **C/A** y **P(Y)**. Abajo del todo se encuentra el generador que forma el mensaje de navegación. Los bloques que generan los códigos y el que genera el mensaje de navegación se encuentran sincronizados a partir de una señal X_1 proporcionada por el generador de códigos **P(Y)**. El bloque *C/A code generator* equivale al diagrama visto en la figura 2.14.

Una vez formados los códigos, estos se combinan con los datos de navegación mediante una suma módulo-2. Si se trata de secuencias binarias representadas con 1s y 0s, se utilizará una OR exclusiva, si, en cambio, la secuencia binaria se representa de forma polar sin retorno a cero (1s y -1s) se podría utilizar una multiplicación ordinaria. (Ver Tabla 2.4)

OR exclusiva			Multiplicación ordinaria		
Input	Input	Output	Input	Input	Output
0	0	0	-1	-1	1
0	1	1	-1	1	-1
1	0	1	1	-1	-1
1	1	0	1	1	1

Tabla 2.4: Otuput de una OR exclusiva y de una multiplicación ordinaria

Las combinaciones de los códigos **C/A** y **P(Y)** junto con los datos de navegación pasan a los moduladores **BPSK** de la portadora L1 y es aquí donde las señales son moduladas (Ver Fig. 2.22). Hay que notar que en la portadora L1 aparecen ambos códigos, aunque uno se encuentra modulado en en fase y el otro en cuadratura, es por eso que hay un desfasado de 90^{o} entre los dos códigos. Finalmente, después de de la parte donde el **P(Y)** es atenuado 3 dB, estas dos señales L1 se suman para formar la señal definitiva. Aunque la señal resultante se encuentre formada por el código encriptado **P(Y)**, el llamado **SPS** está basado solamente por la señal formada a partir del código **C/A**.

La señal transmitida por un satélite k puede definirse como:

$$\begin{aligned} s^k(t) = \sqrt{2P_C}(C^k(t) \oplus Dk(t))cos(2\pi f_{L1}t) \\ + \sqrt{2P_{PL1}}(P^k(t) \oplus Dk(t))sin(2\pi f_{L1}t) \\ + \sqrt{2P_{PL2}}(P^k(t) \oplus Dk(t))sin(2\pi f_{L2}t) \end{aligned} \quad (2.16)$$

donde P_C, P_{PL1} y P_{PL2} son la potencia de las señales los códigos **C/A** o **P**, C^k es el código **C/A** asignado al satélite k, P^k es el código **P(Y)** asociado al satélite k, D^k son los datos de navegación, y f_{L1} y f_{L2} son las frecuencia portadoras de L1 y L2 respectivamente.

Como solamente vamos a trabajar con el código **C/A**, que sólo aparece en la banda L1, los factores afectados por el código **P** y la banda L2 se anulan, dejando nuestra señal simplificada de la siguiente forma:

$$s^k(t) = \sqrt{2P_C}(C^k(t) \oplus Dk(t))cos(2\pi f_{L1}t) \quad (2.17)$$

La figura 2.22 aparece un ejemplo de como sería el resultado de modular con una modulación **BPSK** una señal binaria aleatoria que equivaldría al mensaje de navegación (D), junto a un código **C/A** (C) real sobre una portadora L1 (Carrier). En esa figura

se puede apreciar como la suma módulo-2 entre los datos de navegación y el código de ensanchado ($C \oplus D$) resulta en una inversión del código de ensanchado durante el valor 1 de los datos de navegación.

2.6 Desplazamiento Doppler

El efecto Doppler se puede describir como un fenómeno que modifica la frecuencia de una onda electromagnética debido al movimiento del transmisor respecto su receptor, o viceversa. En el caso en el que el transmisor se aproxima al receptor, cada onda tarda un poco menos de tiempo que la anterior en llegar al receptor, esto provoca que se reduzca el tiempo de llegada de las ondas al receptor lo que causa un aumento de la frecuencia. En el caso opuesto, las ondas se separarían provocando que la frecuencia aumente.

Como el sistema **GPS** no es un sistema geoestacionario, no se encuentra exento de este fenómeno, es más, resulta ser un parámetro importante en el bloque de adquisición de los receptores **GPS** en los que se debe estimar esta variación de la frecuencia.

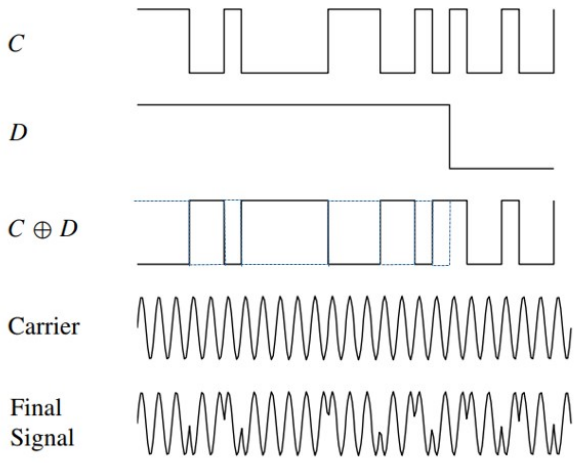

Figura 2.22: El efecto de la modulación BPSK de la portadora L1 con el código C/A y los datos de navegación

Esta variación de la frecuencia producida por el efecto Doppler puede ser mayor o menor que el valor de la frecuencia intermedia definido por el receptor y varía según se encuentra o no en movimiento. Diferentes estudios han determinado el posible rango del desplazamiento de la frecuencia para un receptor estático y para uno en movimiento, siendo estos rangos de $\pm 5khz$ para el receptor estático y $\pm 10kHz$ para el receptor en movimiento.

El Doppler no sólo tiene efecto en la frecuencia portadora de la señal **GPS**, también afecta al código **C/A** pero en menor medida debido al bajo *chip rate* o frecuencia de chip. La frecuencia de chip del código **C/A** es de $1{,}023MHz$, que respecto a la frecuencia portadora L1 es 1540 veces menor ($1575{,}42MHz/1{,}023MHz$). A pesar de esto, el Doppler que afecta al código **C/A** puede causar un desalineación entre el código recibido y el creado en el receptor durante el bloque de adquisición.

Capítulo 3

Receptor digital GPS

Una vez explicado como funciona el sistema **GPS** es el momento de centrarnos en explicar los diferentes bloques que forman el receptor del sistema **GPS**.

En la figura 3.1 se observa un diagrama genérico de un receptor de **GPS**. En él se puede ver como el receptor esta formado por diferentes módulos que vamos a explicar en este capitulo, profundizando solamente en el bloque de la antena/*front-end* y el de adquisición, debido a que en este trabajo nos hemos centrado en esos bloques.

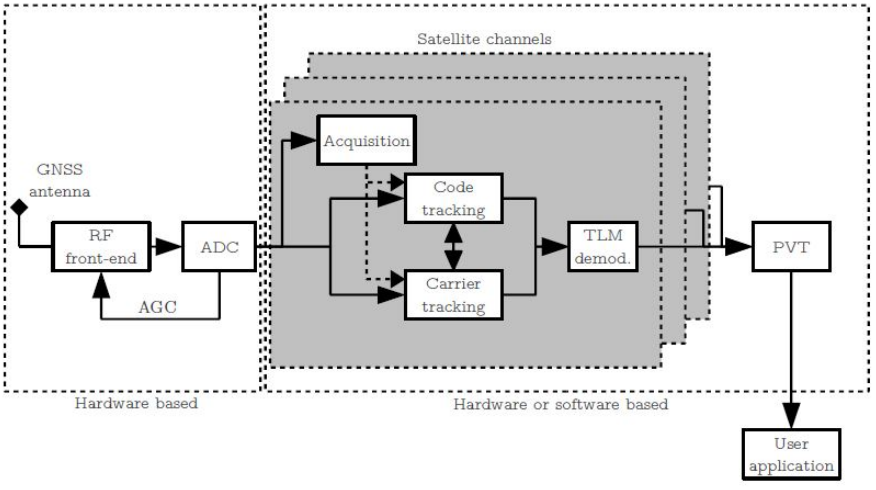

Figura 3.1: Esquema de un receptor GPS

3.1 Antena/Front-end

El primer bloque que podemos encontrar en el diagrama 3.1 es el llamado *Hardware based*, este bloque se encuentra formado por la antena y el llamado *RF front end*, estos dos bloques son los encargados de captar la señal **GPS** analógica y digitizarla.

El proceso de captura de la señal empieza cuando una señal **GPS** incide sobre la antena y esto induce un voltaje en dicha antena. Este voltaje es muy débil, es el correspondiente a una señal de -160 dBW y con una portadora de 1575.42 MHz, la potencia es tan baja que la señal recibida se encuentra por debajo del umbral del ruido térmico. Esto es algo único en las transmisiones por radio y es por esto por lo que utilizamos **CDMA** *spread spectrum*. Además, que la potencia de la señal sea tan baja,

influye en el diseño del front-end haciendo que este se base en el nivel del ruido.

Con el fin de realizar la captura a tiempo real de la señal **GPS**, planteada en este trabajo, la antena y el *front-end* juegan un papel fundamental, es por esto que hemos adquirido una antena **GPS** a la frecuencia de la portadora L1 y como front-end se a utilizado el **SDR** *HackRF*.

Mediante estos dos componentes y con el uso del software **GNSS-SDR** se ha conseguido capturar una señal **GPS** real que será procesada con el fin de obtener el resultado del bloque de adquisición. El montaje para la captura de la señal es el que podemos ver en la figura 3.2.

Figura 3.2: Montaje del SDR con la antena para realizar la captura de la señal

3.1.1 Antena

Normalmente la antena no es considerada como parte del *front-end*, pero como se trata del primer elemento del receptor y es el encargado de captar la señal que los satélites envían, en esta sección vamos a analizar dicho componente.

La antena que se utiliza en los receptores **GPS** se diseña para que a partir de la señal **GPS** que se propaga a la frecuencia L1 o 1575.42 MHz se induzca un voltaje en la antena. Probablemente, dos de los diseños más populares de las antenas para **GPS** son las antenas parche y las hélices, así que vamos a analizar las propiedades más importantes de estas antenas.

El primer parámetro de diseño es el **VSWR** o *Voltage Standing Wave Ratio* que indica cuanta de la potencia incidente se absorberá y cuanta se reflejará. El **VSWR** suele ser del orden 2.0:1 y equivale a un 10 % de la señal reflejada, un 90 % absorbida.

La polarización de estas antenas es, normalmente, circular a mano derecha (**RHCP**

o *Rigth Hand Circular Polarization*). Esta polarización no es arbitraria ya que uno de los efectos más perjudiciales, como es el efecto multicamino, se consigue mitigar con este tipo de polarización. Cuando una señal con polarización **RHCP** rebota en un objeto, esta se invierte pasando a ser una señal con polarización **LHCP** o *Left Hand Circular Polarization*.

En cuanto al diagrama de radiación de este tipo de antenas, sería de suponer que se quisiese, idealmente, un diagrama de radiación omnidireccional, en cambio, este no es el caso. Si el diagrama de radiación fuese omnidireccional, la antena captaría señales que no tendrían ningún sentido con el propósito de la aplicación, es por eso que el diagrama que tienen es casi-omnidireccional, en el que solamente se cubre todas las direcciones del espacio para una elevación positiva y para todos los valores de azimut.

Figura 3.3: Antena GPS utilizada

Otros dos parámetros muy importantes en las antenas son la ganancia y el factor de ruido, pues al tratarse del primer elemento, tendrán mucha influencia en el factor de ruido del sistema (Fórmula de Friis (3.1)). Por lo tanto, desearemos tener un factor de ruido muy bajo y una ganancia lo más alta posible.

$$F_{sistema} = F_1 + \frac{F_2 - 1}{G_1} + \frac{F_3 - 1}{G_1 G_2} + \ ... \ + \frac{F_n - 1}{G_1 G_2 ... G_{n-1}} \tag{3.1}$$

Teniendo en cuenta los parámetros antes definidos, podemos analizar, a partir de sus características, si nuestra antena es apta para este propósito:

- Conector SMA macho activo a antena GPS
- Longitud de 3m
- Frecuencia de 1575.42MHz ± 3MHz
- Ganancia LNA (sin cable) 25dB
- Factor de ruido menos de 1.5dB
- VSWSR menos de 2.0
- Corriente de 10mA máximo
- Voltaje 3 - 5V

3.1.2 Front-end/HackRF

El bloque del *front-end* es el encargado de amplificar, filtrar y convertir a una frecuencia intermedia o **IF** la señal analógica obtenida por la antena. Una vez trabajada la señal analógica un **ADC** (Analog-to-Digital Converter) es el encargado de digitalizar esta señal.

El proceso anterior lo realizará el **SDR** *HackRF*, al cual le pasarán los parámetros para su funcionamiento el *software* GNSS-SDR.

El *HackRF One* es un **SDR** capaz de transmitir y recibir señal radio desde 1 MHz hasta 6 GHz, cuyas características son las siguientes:

- Frecuencia de operación desde 1 MHz hasta 6 GHz.
- Comunicación *half-duplex*
- Máximo de hasta 20 millones de muestras por segundo.
- Muestras 8-bit I y 8-bit Q.
- Compatible con GNURadio.
- *Software* reconfigurable, ganancia en RX y TX y el filtro bandabase.
- USB-powered.
- Compatible con USB 2.0
- Alimenta al puerto de la antena (50 mA a 3,3 V)

3.2 Adquisición

El bloque de adquisición de un receptor **GPS** es el encargado de detectar la presencia, o no, de la señal emitida por un satélite **GPS**. Este bloque se explicará más profundamente en el Capítulo 4 de este documento.

Como hemos dicho la función de este bloque es identificar si de la señal recibida

$$x_{IN}[k] = A(t)s_T(t-\tau(t))e^{j(2\pi f_D(t)t+\phi(t))}|_{t=kT_s} + n(t)|_{t=kT_s} \tag{3.2}$$

se puede identificar al satélite que la envió y obtener una aproximación del retraso del código (τ) y del desplazamiento Doppler (f_D) para facilitárselos al bloque de tracking.

De la ecuación (3.2) $A(t)$ es la amplitud dela señal, $s(t)$ es la señal bandabase compleja obtenida por el *front-end* filtrada, $f_D(t)$ es el desplazamiento Doppler variante con el tiempo, $\phi(t)$ es el desplazamiento en fase de la portadora variante con el tiempo, T_s es el periodo de muestreo y $n(t)$ corresponde con el ruido **AWGN** además de otros efectos indeseados como pueden ser las interferencias o las perdidas debidas al multicamino.

3.3 Tracking

El bloque del tracking del receptor es el encargado de precisar los dos parámetros obtenidos en el bloque de adquisición, el retraso del código y el desplazamiento Doppler, ya que estos dos parámetros pueden experimentar variaciones a lo largo del tiempo. Estos serán monitorizados a la vez por dos arquitecturas distintas: el retraso del código por el **DLL** o *Delay Lock Loop* y la frecuencia Doppler por el **PLL** o *Phase Lock Loop*.

El funcionamiento de estas dos arquitecturas, el **DLL** y el **PLL**, son las llamadas arquitecturas en lazo cerrado, las cuales se usan para estimar de forma precisa los parámetros de interés a base de comparar una señal de referencia con una réplica local.

Esta comparación generará una señal de error, que será la que se utilizará para generar una nueva señal réplica para la siguiente iteración. El objetivo es que el error vaya teniendo a cero, pues esto significará que tanto la señal de referencia como la réplica se encuentran alineadas. El resultado de este bloque serán la estimación del retraso del código y los *telemetry symbols*

3.4 Telemetry

El objetivo de este bloque es demodular los *telemetry symbols* obtenidos en el bloque de tracking. Este bloque será el encargado de decodificar el mensaje de navegación, de la cual pudimos ver la estructura en la sección *2.4.2*.

3.5 PVT

El **PVT** o *Position, Velocity and Time* se encarga de obtener, a partir de la información obtenida en el bloque de *Telemetry*, la posición del receptor haciendo uso del sistema de ecuaciones (2.2). Como ya se ha comentado, para resolver el sistema de ecuaciones (2.2) es necesario conocer la posición exacta de cada satélite que ha enviado la señal **GPS** y el error de reloj, esta información es la que el bloque *Telemetry* obtiene.

Para realizar la tarea anterior, en este bloque se realizan diversos cambios de formato, tanto para el tiempo como para las distancias, ya que el sistema **GPS** utiliza un sistema de tiempos propio llamada *GPS System Time* y las distancias depende del tipo de modelado de la Tierra escogido.

Capítulo 4

Adquisición

El propósito del bloque de adquisición de un receptor es identificar todos los satélites visibles que tiene. Si un satélite es visible, el bloque de adquisición debe determinar las dos siguientes propiedades de la señal recibida:

- *Frecuencia*. Es necesario obtener el valor de la frecuencia de la señal de un satélite especifico, pues como hemos visto en el apartado *2.6*, la señal se encuentra afectada por el efecto Doppler, por lo que la frecuencia a la que se recibe la señal no corresponderá al valor nominal de la portadora L1. En el caso de una transformación en frecuencia, el valor de la frecuencia de la señal **GPS** en la portadora L1 corresponde a su valor de frecuencia intermedia (**IF**).
- *Code Phase*. El *Code Phase* o retraso del código de la señal **GPS** es un parámetro que indica en que punto de la señal empieza el código **C/A**, como hemos visto en el apartado *2.4.3*, en las propiedades de autocorrelación de los códigos **C/A**, para obtener un máximo de autocorrelación es necesario que los códigos se encuentren perfectamente alineados.

Como hemos mencionado antes, la señal s que llega al receptor es una combinación de la señal de los n satélites visibles:

$$s(t) = s^1(t) + s^2(t) + ... + s^n(t) \tag{4.1}$$

Cuando se adquiere el satélite k, la señal s se multiplica por el código **C/A**, generado localmente, perteneciente a ese satélite k. Mediante este proceso la señal de los otros satélites se elimina debido a que la correlación cruzada entre el código **C/A** de un satélite y el de los demás es prácticamente cero. Para evitar eliminar la señal deseada del satélite k el código **C/A** debe estar perfectamente alineado en tiempo.

Una vez eliminado el código **C/A** la señal se debe multiplicar con una portadora, también generada de forma local, para eliminar la portadora de la señal de entrada. Esta portadora generada de forma local he de ser muy parecida a la de la portadora de la señal recibida, que se encontrará afectada por el efecto Doppler.

El proceso de adquisición trabaja como un proceso de búsqueda de los dos parámetros antes definidos. Para cada uno de los valores diferentes de frecuencia se prueban los 1023 *Code Phases* diferentes. Cuando se han probado todas las posibilidades para todas las frecuencia y todos los retrasos del código **C/A** se realiza la búsqueda del valor máximo, si este valor supera un determinado umbral el satélite se adquiere con su correspondiente frecuencia y *Code Phase*.

4.1 Algoritmos de adquisición

Para implementar la adquisición antes descrita se van a plantear los siguientes algoritmos cuya finalidad será identificar de que satélite/s pertenece la señal recibida y obtener el valor de la frecuencia y el retraso del código **C/A** de esta señal.

4.1.1 Serial Search Acquisition

El primer algoritmo que vamos a explicar es el *serial search acquisition*, este algoritmo se utiliza a menudo en sistemas **CDMA**. En la figura 4.1 se encuentra el diagrama de bloques correspondiente a este algoritmo.

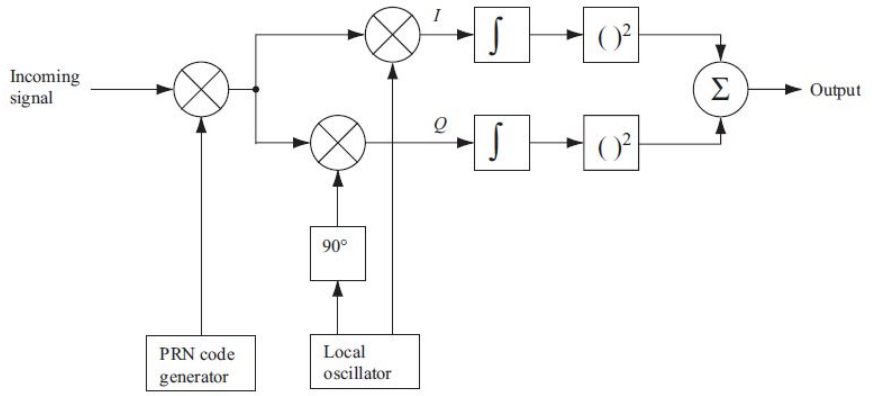

Figura 4.1: Diagrama de bloques del Serial Search Acquisition

Tal y como se puede ver en el diagrama de bloques, la señal de entrada se multiplicará por los códigos **PRN** y por una portadora generados localmente. El generador de códigos **PRN** generará un código perteneciente a un satélite en especifico, con un retraso o *Code Phase* de 0 a 1022 chips. La señal de entrada y este **PRN** se multiplican entre sí y dicho resultado se multiplica por una frecuencia portadora. Esta última multiplicación genera la señal en fase (**I**), y la misma multiplicación pero con la portadora desfasada 90^{o} genera la señal en cuadratura (**Q**). Ambas señales resultantes son integradas durante 1 ms, duración de 1 periodo de código **C/A**, para finalmente elevar al cuadrado y sumar el resultado.

Idealmente, la potencia de la señal debería estar ubicada en la parte **I** de la señal recibida, ya que el código **C/A** sólo se modula en fase. En este caso, sin embargo, la señal en fase generada en el satélite no tiene porque corresponder a la demodulada, esto es debido a que la fase de la señal recibida es desconocida. Así que para tener certeza de que la señal ha sido detectada es necesario investigar las señales en **I** y en **Q**. La salida es un valor de la correlación entre la señal de entrada y la generada de forma local, si este valor excede un umbral predefinido el satélite ha sido adquirido y, tanto la frecuencia como el retraso del código son correctos y los podremos usar en los algoritmos de *tracking*.

Este algoritmo realiza dos barridos diferentes: un barrido en frecuencia para todas las posibles frecuencias de entrada debidas al Doppler (**IF** $\pm$ 10 KHz para un receptor en movimiento) en *steps* o pasos de 500Hz y un barrido a través de los 1023 desfases de código posibles. Todo esto suma un total de

$$\underbrace{1023}_{code\ phases}\ \underbrace{(2\frac{10000}{500}+1)}_{frecuencias} = 1023{\cdot}41 = 41943\ combinaciones \tag{4.2}$$

Naturalmente el número de combinaciones posibles es muy elevado, por lo que este suele ser el principal defecto de este algoritmo.

4.1.2 Parallel Frequency Space Search Acquisition

El anterior algoritmo de adquisición consume mucho tiempo al tener que buscar de manera secuencial todos los posibles valores para dos parámetros, la frecuencia y el retraso del código. Si alguno de estos dos parámetros se pudiese eliminar de la búsqueda o implementarlo en paralelo, el rendimiento del proceso mejoraría de forma significativa.

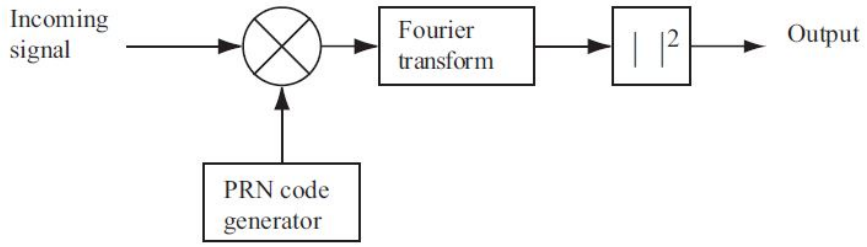

Figura 4.2: Diagrama de bloques del Parallel Frequency Space Search Acquisition

Tal como su nombre indica, este método implementa en paralelo la búsqueda de uno de estos dos parámetros, en concreto la frecuencia. Tal como se ve en el diagrama de bloques de la figura 4.2 este algoritmo se diferencia del anterior en el que se realiza una transformada de Fourier para pasar la señal del dominio del tiempo al de la frecuencia.

La señal de entrada es multiplicada por el **PRN** generado de un satélite en especifico y con un retraso de entre 0 y 1022 chips. La señal resultante se transforma al dominio de la frecuencia mediante una transformada de Fourier, esta transformada se puede implementar mediante una **DFT** (Discrete Fourier Transform) o mediante una **FFT** (Fast Fourier Transform).

En este algoritmo, el resultado de la multiplicación es pasado al bloque que realiza la transformación en frecuencia (la **DFT** o **FFT**), la representación de la señal de la salida de este bloque en el dominio de la frecuencia mostrará un pico. La frecuencia de este pico corresponderá con la frecuencia de la portadora de la señal de entrada.

Después de transformar la señal al dominio de la frecuencia mediante la **FFT** dicha señal pasa a ser una señal compleja. Si el código **PRN** estaba correctamente alineado con el código de la señal de entrada, la señal a la salida de la **FFT** tendrá un pico en el mismo valor de frecuencia que la portadora de la señal de entrada. Para encontrar este pico se hace el valor absoluto al cuadrado de todas las componentes y se busca el valor máximo, si este valor supero un umbral hemos adquirido la señal del satélite.

La precisión con la que se determina esta frecuencia depende de la longitud de la **DFT**. La longitud corresponde con el número de muestras analizadas en la señal de entrada. Si analizamos 1 ms de la señal de entrada, el número de muestras se puede obtener como $f_s/1000$. Si, por ejemplo, la frecuencia de muestreo f_s es 10 MHz, el numero de muestras es N = 10000.

Con una **DFT** de tamaño 10000, las primeras $N/2$ muestras de salida representan a

las frecuencias desde 0 hasta $\frac{f_s}{2}$ Hz. Esto significa que la resolución en frecuencia sería

$$\triangle f = \frac{f_s/2}{N/2} = \frac{f_s}{N} \tag{4.3}$$

Siguiendo el anterior ejemplo, con una $f_s = 10$ MHz la resolución en frecuencia resultante sería

$$\triangle f = \frac{10MHz}{10000} = 1kHz \tag{4.4}$$

En este caso la precisión sería de 1kHz, dicha precisión mejora si la comparamos con la precisión del anterior algoritmo de solamente 500Hz (el valor del *step* o paso).

Mientras que el *serial search acquisition* busca de entre los posibles valores de retraso del código y de la frecuencia portadora, este método solamente prueba con los 1023 diferentes retrasos de código. El coste computacional de este método reside en la transformada de Fourier, dependiendo de como se implemente dicha transformada sería posible realizar una implementación más rápida de este algoritmo que del anterior.

4.1.3 Paralel Code Phase Search Acquisition

Tal como vemos en la ecuación (4.2), el número de pasos para la búsqueda del retraso del código es bastante mayor al de la frecuencia (1023 comparado con 41). El anterior algoritmo elimina la necesidad de buscar a través de las 41 posibles frecuencias, si se pudiese eliminar la necesidad de buscar a través de los distintos retrasos del código, solamente se necesitaría buscar a través de las 41 posibles frecuencias. Este algoritmo utilizará esta idea.

La meta de la adquisición es realizar una correlación entre la señal de entrada y el código **PRN**. Si en lugar de multiplicar la señal de entrada con un **PRN** con 1023 retardos diferentes, como se hacía en el *serial search acquisition*, se realiza una correlación cruzada circular entre la señal de entrada y el código **PRN** sin desfase, se elimina la necesidad de buscar en los 1023 retrasos posibles. En este algoritmo se describirá la relación entre hacer una correlación circular a partir de una transformada de Fourier.

La transformada discreta de Fourier de las secuencias finitas $x(n)$ y $y(n)$, ambas con una longitud de N, se definan como

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N} \quad \text{y} \quad Y(k) = \sum_{n=0}^{N-1} y(n)e^{-j2\pi kn/N} \tag{4.5}$$

La autocorrelación circular entre dos secuencias finitas $x(n)$ y $y(n)$, ambas con una longitud de N y con repeticiones periódicas, se define como

$$z(n) = \frac{1}{N}\sum_{m=0}^{N-1} x(m)y(m+n) = \frac{1}{N}\sum_{n=0}^{N-1} x(-m)y(m-n) \tag{4.6}$$

Para una mayor simplicidad omitiremos el factor de escala $\frac{1}{N}$.

La transformada discreta de Fourier de tamaño N de $z(n)$ se puede expresar como

$$Z(k) = \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} x(-m)y(m-n)e-j2\pi kn/N$$
$$= \sum_{m=0}^{N-1} x(m)e^{j2\pi km/N} \sum_{n=0}^{N-1} y(m+n)e^{-j2\pi k(m+n)/N} = X^*(k)Y(k) \quad (4.7)$$

donde $X^*(k)$ es el complejo conjugado de $X(k)$.

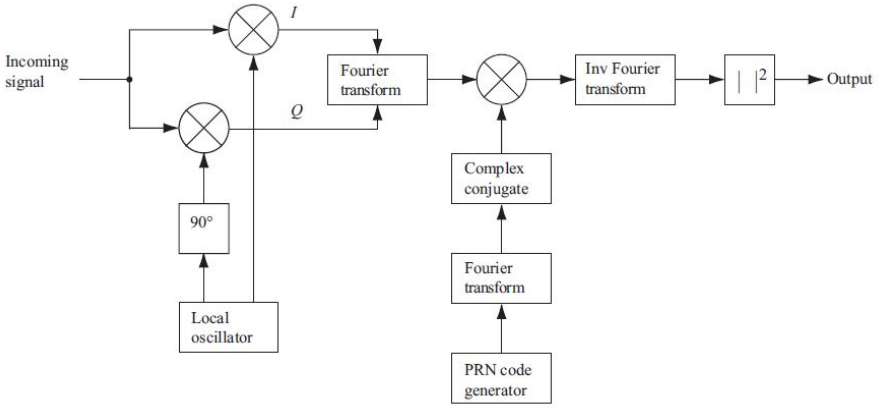

Figura 4.3: Diagrama de bloques del Parallel Code Phase Search Acquisition

En la figura 4.3 se representa el diagrama de bloques de este algoritmo. La señal de entrada se multiplica por la portadora generada localemnte, esta multiplicación genera la señal en **I**, mientras que la misma multiplicación con la portadora desfasada 90^o genera la señal en **Q**. Las señales **I** y **Q** se combinan formando $x(n) = I(n) + jQ(n)$, será esta $x(n)$ la que pase por el bloque que realiza la transformada de Fourier.

El **PRN** generado también se transforma al dominio de la frecuencia y al resultado se le hace el complejo conjugado.

La salida del bloque que hace la transformada de Fourier es multiplica con del código **PRN**, este resultado es transformado al dominio del tiempo mediante una transformada inversa de Fourier. El valor absoluto de la salida de la transformada inversa de Fourier representa la correlación entre la señal de entrada y el código **PRN**. Si aparece un pico en la correlación, este marcará el desfase del código **PRN** de la señal de entrada.

Comparado con los anteriores algoritmos en este solamente se busca en las 41 posibles portadoras diferentes, aunque la carga computacional reside en ambas transformadas de Fourier (la directa y la inversa), ya que se realiza cada trasnformada para cada una de las posibles 41 frecuencias.

La precisión de este algoritmo, al igual que el *serial search acquisition*, reside en el *step* o paso dado para formar las distintas posibles portadoras afectadas por el Doppler. En cambio, el retraso del código **PRN** tiene mayor precisión que en los dos algoritmos anteriores, pues nos da el valor de la correlación para cada retraso de código. Si por ejemplo la f_s es igual a 10 MHz, el códiog **PRN** muestreado tendrá 10000 muestras, así que la precisión del retraso del código tendrá 10000 valores diferentes en lugar de 1023.

4.1.4 Tamaño de los datos a analizar

En los anteriores apartados nos hemos centrado en conocer los diferentes algoritmos que se utilizan en la etapa de adquisición de la señal **GPS**, pero al igual que es importante conocer estos como funcionan estos algoritmos, es necesario saber el tamaño de la señal de entrada de estos, pues dependiendo de ciertos criterios este valor puede variar.

El valor del tamaño de los datos de entrada acaba siendo una solución de compromiso entre tres problemas que vamos a describir a continuación:

- El primer inconveniente lo encontramos en la transición de los bits del mensaje de navegación. Si durante la adquisición de una señal a tiempo real se produce una transición de bits del mensaje de navegación, sucede que el pico obtenido por la correlación no será máximo y esto pueda causar problemas tales como: una mala estimación de la frecuencia de la señal y del retraso del código, así como la obtención de un falso negativo. Es por esto que el tamaño de datos a analizar debe asegure que no se produzca ninguna transición para obtener el óptimo funcionamiento de los algoritmos antes analizados.
- El segundo inconveniente es seleccionar un tamaño de datos que permita una adquisición positiva. Este inconveniente es debido a que la posibilidad de adquirir correctamente un satélite aumenta con el tamaño de los datos analizados.
- El tercer y último inconveniente es el coste computacional. Como es de esperar si se analiza un mayor número de muestras de la señal **GPS** el coste computacional del algoritmo aumentará. Este inconveniente es la contrpartida del anterior, pues si se aumenta el tamaño de los datos a analizar aumenta la probabilidad de detectar al satélite en cuestión, pero también aumenta el coste computacional del algoritmo

Como hemos dicho el valor del tamaño de los datos a analizar acaba siendo una solución de compromiso entre los tres inconvenientes anteriores, siendo este valor el de 1 ms de la señal **GPS**. Este valor corresponde exactamente a la longitud de un periodo del código **C/A** lo que simplifica el algoritmo al no tener que replicar los códigos. El tamaño de los datos difícilmente puede ser menor, ya que involucraría correlaciones con códigos incompletos.

4.2 Implementaciones

Durante todo el trabajo se ha analizado como funciona el sistema **GPS**, sus características más importantes en cuanto a funcionamiento, pero sobretodo como funciona y como esta hecha la señal que los satélites transmiten. Una vez entendido el sistema, nos hemos centrado en conocer el funcionamiento de un receptor centrándonos, básicamente, en las etapas del *front-end* y de la adquisición. Así que, finalmente, sólo queda poner en práctica todo lo visto a lo largo de este documento para obtener diferentes adquisiciones para distintas señales **GPS**.

Como ya se ha explicado al principio de este capitulo, el objetivo de la etapa de adquisición es estimar que satélites son visibles por el usuario y de aquellos que lo son obtener dos parámetros clave como son el desplazamiento Doppler y el retraso del código. Para ello se realiza una búsqueda en tiempo y en frecuencia mediante la generación de una señal réplica a la captada a la salida del front-end.

Para realizar las adquisiciones de las distintas señales se ha empleado el algoritmo **PCPS** cuyo proceso de diseño se puede ver en el flujograma de la figura 4.4.

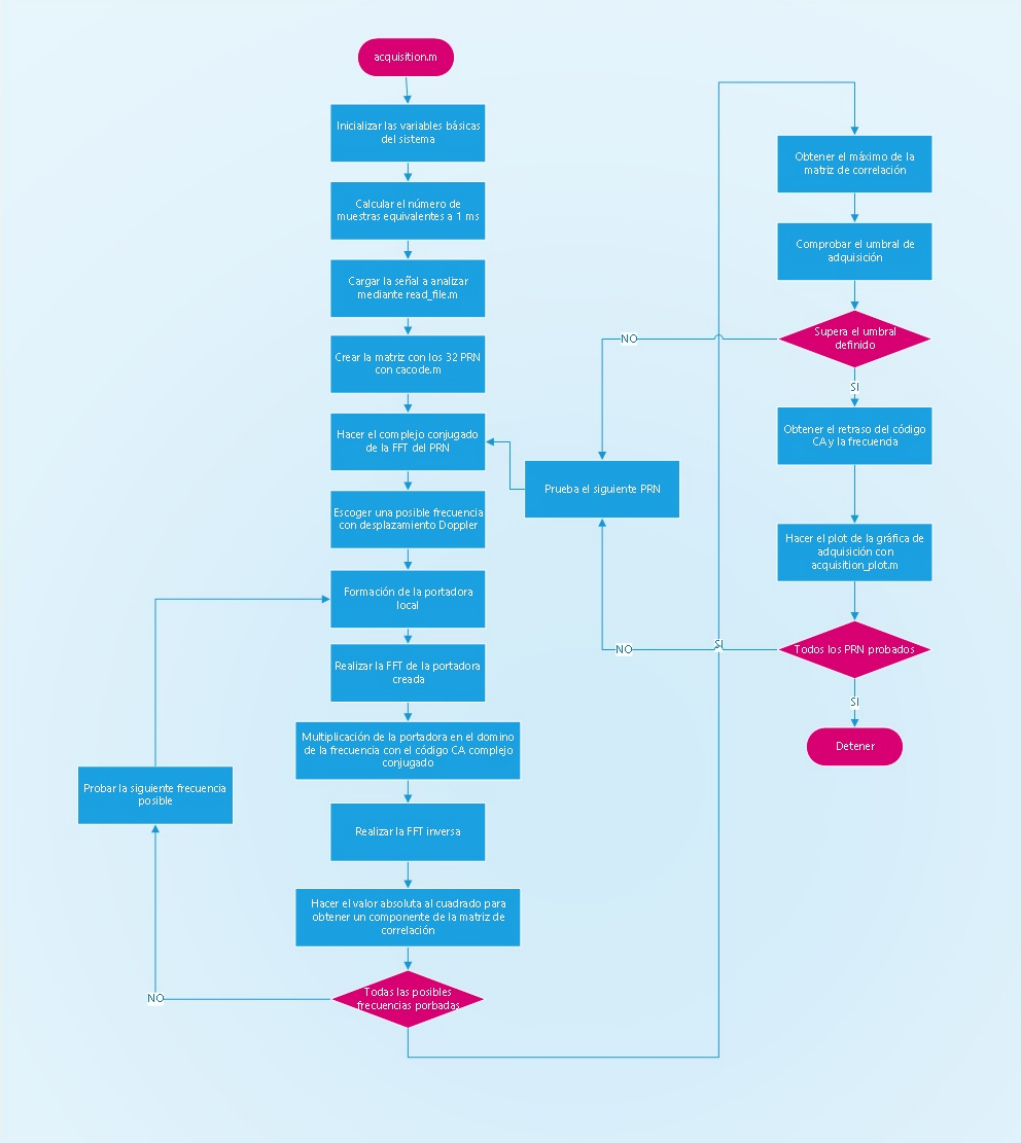

Figura 4.4: Flujograma del algoritmo PCPS

4.2.1 Muestreo de la señal

El bloque de adquisición obtiene la señal **GPS**, esta señal es un flujo contante de muestras listas para que sean procesadas. Por lo que nuestro primer paso, y que será común en las diferentes implementaciones realizadas, es el de acondicionar el número de muestras del código **C/A**.

Antes de llegar a la etapa de adquisición, la señal **GPS** es muestreada a una cierta frecuencia, f_s. Por lo tanto, lo primero que debemos obtener es el número de muestras que tiene un chip del código **C/A** para cierta frecuencia f_s

$$m_{Chip}[\text{Muestras}] = f_s[\text{Muestras/s}]\ T_{chip}[\text{s}] \tag{4.8}$$

Recordando que el T_{chip} es

$$T_{chip} = \frac{1}{f_{chip}} = \frac{1}{1{,}023MHz} = 977{,}52\mu s \approx 1\mu s \tag{4.9}$$

Suponiendo una frecuencia de muestreo de $6MHz$

$$m_{Chip} = 6MHz\ 1\mu s = 5{,}865 \text{ Muestras por chip} \tag{4.10}$$

De esto se concluye que para esa frecuencia de muestreo, cada chip del código **C/A** equivale a 5,865 muestras, esto quiere decir que, muestrear la señal a cierta frecuencia hace que cada chip del código **C/A** que originalmente se encontraba representado por 1 muestra se encuentre, ahora, representado por más. Por lo tanto, se debe acondicionar el código **C/A** que vamos a replicar para que cada chip tenga 5,865 muestras, o lo que es lo mismo, que la longitud total del código **C/A** pase de 1023 a

$$Long_{1msCA} = N_{chips}\ m_{Chip} = 1023\ 5{,}865 = 6000 \text{ Muestras por código C/A} \tag{4.11}$$

Este paso se realiza automáticamente en la función *cacode.m* del Anexo 1. Esta función se usará en todas las simulaciones para hacer el cálculo del código **C/A** de los diferentes satélites dependiendo de la frecuencia de muestreo que usemos en cada simulación.

Aplicando la función antes nombrado para generar el código **C/A** de un satélite para una frecuencia de muestreo de $6MHz$ se obtendría la siguiente señal:

$$CA = [CA(1)\ CA(1)\ CA(1)\ CA(1)\ CA(1)\ CA(1)\ CA(2)\ CA(2)\ ...\ CA(1023)] \tag{4.12}$$

En media a cada chip le corresponderían 5.865 muestras y la señal **C/A** generada locamente en el receptor tendría una longitud de 6000 muestras.

4.2.2 Primera simulación: señal ejemplo

En esta primera simulación se ha realizado la adquisición de una señal ejemplo preparada para tal propósito. Esta señal, de duración 1 ms y muestreada a 11.999 MHz, se encuentra formada por los códigos de los satélites 1 y 10.

Para realizar la adquisición de esta señal se ha implementado el algoritmo **PCPS** explicado en el punto *4.1.3*.

La función realizada para la adquisición se puede ver en el Anexo 1, pero para una mayor compresión del algoritmo y del código, en la figura 4.5 se puede ver el nombre de las variables dadas en cada una de las partes del diagrama de bloques.

En este algoritmo para cada uno de los 32 códigos **C/A** posibles se prueban las 41 frecuencias con desviación Doppler distintas, cabe decir que este valor viene explicado en la sección *2.6*, aunque se puede recordar que viene determinado por el posible rango de la desviación en frecuencia que ha podido tener la portadora (dependiendo de si el receptor se encuentra o no e movimiento) dividido entre cierto espaciado, que es el que afecta a la precisión del algoritmo.

$$f = \frac{+10kHz - (-10kHz)}{500} + 1 = 41 \tag{4.13}$$

Con todas las frecuencias probadas para un mismo código, se obtiene la matriz de autocorrelción *u*. Esta matriz, de tamaño 41x1199, contiene la correlación entre la señal de entrada y la señal replica para todos los posibles retrasos del código **C/A** y para todas las posibles frecuencias con una desviación de frecuencia de $\pm 10kHz$ con un paso de 500Hz.

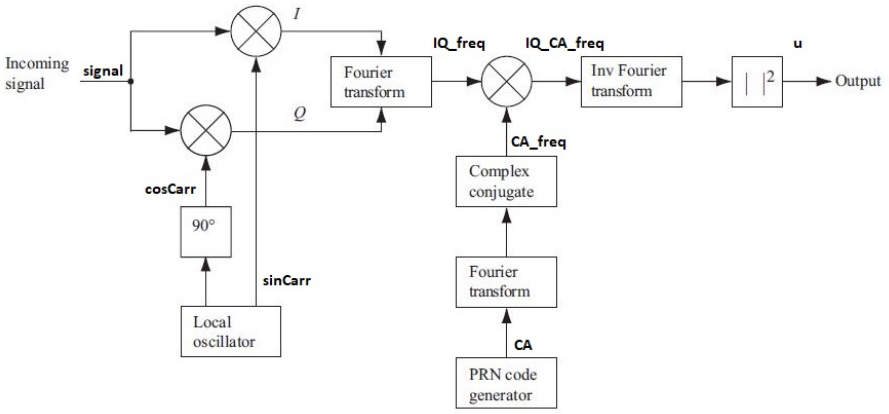

Figura 4.5: Implementación del algoritmo PCPS

Será esta matriz de autocorrelación la que se representará y en la que se podrá observar si aparece, o no, el pico de autocrrelación indicando que, para dicho satélite, hemos adquirido la señal. (Ver Fig. 4.6 y 4.7)

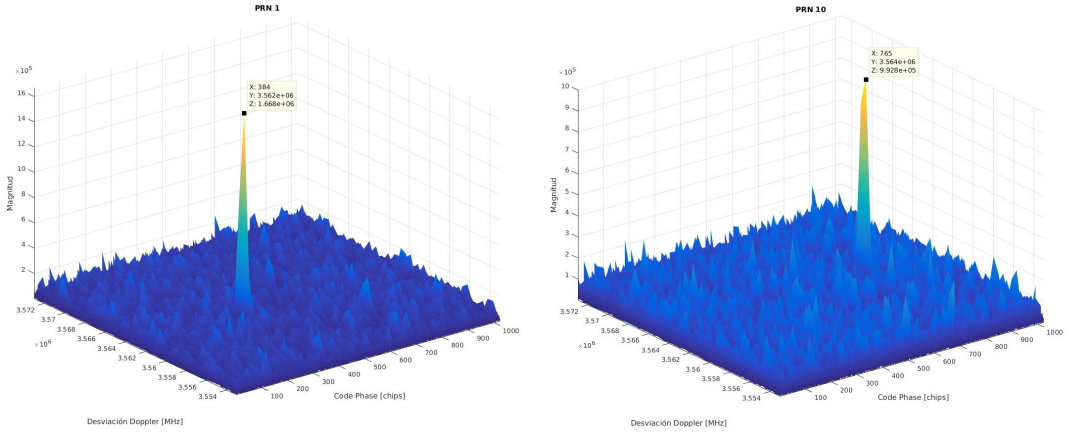

Figura 4.6: Adquisición señal simulada: (a) PRN1 (b) PRN10

Observando las figuras 4.6 y 4.7 vemos como para la figura 4.6 se aprecia un pico de correlación para los **PRN 1 y 10** claramente diferenciado, no como en el **PRN 3** en el que no se puede apreciar dicho pico. Para los dos satélites adquiridos, el 1 y el 10, la posición del máximo de la matriz u, en la que se encuentra el valor del pico, nos da la información de los dos parámetros que queremos obtener, la frecuencia y el retraso del código.

La frecuencia se ve representada en la gráfica en el eje de las y y en él se encuentra el valor de la frecuencia intermedia a la que se encontraba esta señal con un pequeño desfase debido al Doppler. (Ver Fig. 4.6)

Para el retraso del código solamente hay que fijarse en el eje de las x, pero al contrario que en la frecuencia, este valor ha de ser calculado, pues hay que recordar que el tamaño de la matriz u era de 41x11999. Como se vio en la sección *4.1.3* en este algoritmo el retraso del código es más preciso ya que se trabaja con un retraso mayor a los 1023 posibles, en este caso 11999. Así que a partir de la posición y del máximo de la matriz u se obtiene

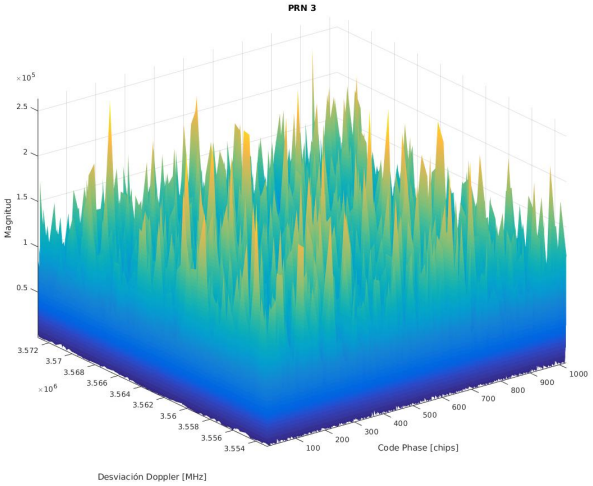

Figura 4.7: Adquisición señal simulada: PRN3

dicho retraso para el valor de 11999

$$CodePhase = y - 1 \tag{4.14}$$

El valor obtenido hay que escalarlo a su valor real de 1023 posibles retrasos

$$CodePhase1023 = CodePhase/f_s \cdot 1{,}023MHz \tag{4.15}$$

4.2.3 Segunda simulación: señal simulada

Para esta segunda simulación se hará uso de la función que simula una señal **GPS** a la entrada del bloque de adquisición (Anexo 1). La señal simulada corresponde a 1 bit de señal, 20 ms, muestreada a 6 MHz a la que se le ha aplicado un retraso del código de valor 4000 y que se le ha añadido ruido producido por el canal y el efecto Doppler y cuyo **PRN** es el del satélite 25.

En este caso, el algoritmo usado en para adquirir dicha señal ha sido distinto al de la primera simulación. Para este ejemplo, hemos utilizado un algoritmo que realiza la correlación a partir de la convolución de la señal **GPS** con la réplica del código **C/A** generada en el receptor. Para ello, se ha trabajado con 2ms de la señal **GPS** simulada y con 1 periodo de la réplica del código **C/A**, 1 ms. Esto se debe a que escogiendo estos tamaños nos aseguramos que dentro de la señal tendremos un periodo del código.

Como hemos dicho, la señal simulada ha sido muestreada a 6 MHz, lo que quiere decir que 1 periodo del código **C/A** tendrá

$$Long_{1msCA} = N_{chips} \cdot m_{Chip} = 1023 \cdot f_s \cdot T_{chip} = 6000 \text{ Muestras} \tag{4.16}$$

Al ser generado por el receptor, este código se encontrará perfectamente alineado, esto quiere decir que la primera muestra del código corresponderá a la primera muestra del primer chip. (Ver Fig. 4.8)

Por otra parte, al utilizar 2 ms de la señal **GPS** estaremos trabajando con 12000 muestras de las que desconocemos en cual de ellas empieza su código **C/A**. (Ver Fig. 4.9)

Como se observa en las figuras 4.8 y 4.9 del ejemplo planteado, la primera muestra del código **C/A** se encuentra retrasada 4 muestras, con lo que la adquisición de esta señal nos debería dar como valor del $CodePhase1023 = 4$. Este es número de muestras que hay

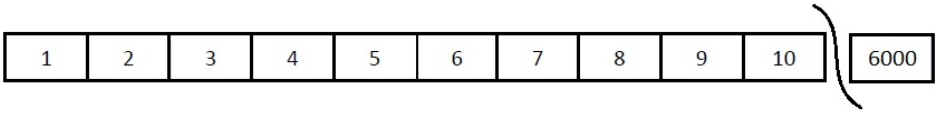

Figura 4.8: Réplica código C/A perfectamente alineado

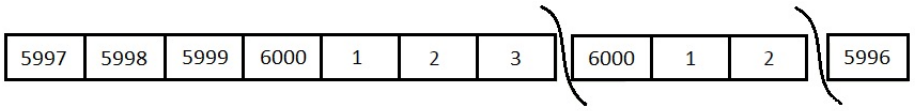

Figura 4.9: Réplica código C/A perfectamente alineado

que recorrer para que la señal replica se encuentre alineada con la señal **GPS**.

Continuado con el ejemplo, para cada una de las frecuencias Doppler se realizará una correlación entre la señal y la réplica del código. Esta correlación se realizará mediante una convolución entre las dos señales. Como lo que se pretende encontrar es el máximo de correlación entre ambas señales, solamente se guardará el resultado del producto de las 6000 muestras centrales de la convolución, se descartarán el resto porque en esos resultado lo más probable es que no encontremos dicho máximo.

Vamos a ver lo explicado anteriormente mediante un ejemplo esquemático de la convolución, en el que la señal superior corresponde a las 12000 muestras de la señal **GPS** y la señal inferior es el código **C/A** girado. Veremos los tres estados distintos de la convolución, en el que la señal sombreada correspondiente al final del estado.

Primer estado
En este primer estado empieza la convolución, en ella se realiza el producto muestra a muestra entre ambas señales hasta que la señal réplica se multiplica por completo con la señal **GPS**. (Ver Fig. 4.10)

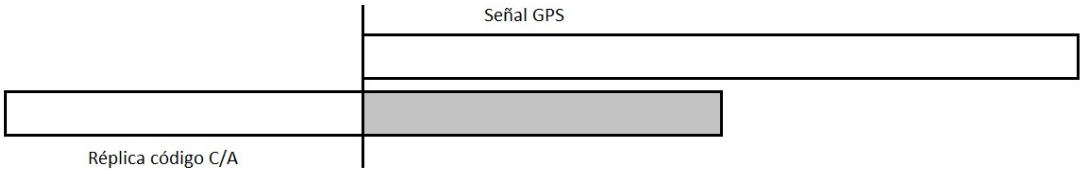

Figura 4.10: Esquema del primer estado de la convolución

Segundo estado
Segundo estado o parte central de la convolución, es en este estado en el que es más probable encontrar el máximo de correlción entre ambas señales. Durante 6000 muestras, ambas señales se multiplicarán muestra a muestra hasta el momento en el que la señal réplica se desplace fuera de la señal **GPS**. (Ver Fig. 4.11)

Tercer estado
Es la parte final de la convolución y equivale al producto de la señal réplica con la señal **GPS** desde que empieza a salir la réplica hasta que esta se encuentra totalmente fuera.

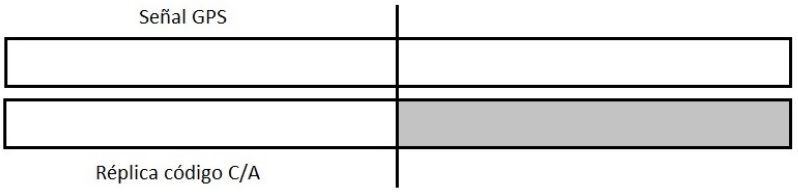

Figura 4.11: Esquema del segundo estado de la convolución

(Ver Fig. 4.12)

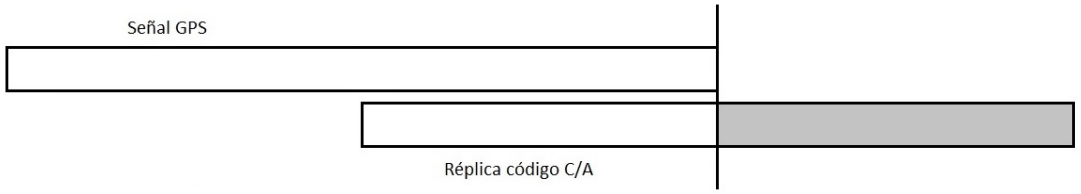

Figura 4.12: Esquema del tercer estado de la convolución

Como ya sabemos el máximo de correlación aparecerá cuando ambas señales se encuentren perfectamente alineadas. Es por esta razón por la que solamente nos interesa el resultado del segundo estado, pues será seguro que dentro de las primeras 6000 muestras de la señal **GPS** (la superior en el esquema) encontraremos el inicio de su código **C/A**.

Una vez se obtiene la matriz de correlación se hará el módulo al cuadrado de cada una de sus celdas y si el valor máximo supera un umbral previamente definido, se podrá confirmar la presencia del satélite.

Los dos parámetros importantes, el desplazamiento Doppler y el retraso del código, se obtendrán de la misma forma que lo explicado para la primera simulación, sección *4.2.2.*

Como se ha dicho al principio de esta sección, para realizar esta simulación se ha creado/simulado una señal **GPS** perteneciente al satélite 25 a la que, entre otras cosas, se le ha aplicado un retraso del código de 4000 muestras. Este valor de retraso del código es el equivalente al código **C/A** muestreado, su valor del retraso para 1023 muestras sería

$$CodePhase = y - 1 = 4000 - 1 = 3999 \tag{4.17}$$

Esta sería la posición y de la matriz de correlación en la que se encontraría el máximo.

$$CodePhase1023 = CodePhase/f_s \; 1{,}023MHz = 3999/6MHz{\cdot}1{,}023MHz = 681{,}83 \tag{4.18}$$

Este sería el valor que se observaría en la gráfica de adquisición. (Ver Fig. 4.13)

4.2.4 Tercera simulación: señal capturada real

En esta, y última simulación, se pretende realizar la adquisición de una señal **GPS** captada previamente mediante la antena y el **SDR** explicados en el capítulo 3. Para ello, se va a utilizar el programa GNSS-SDR. Este programa, entre otras cosas, implementa un receptor digital para **GPS** y **Galileo**, en el que se podrá configurar cada parámetro

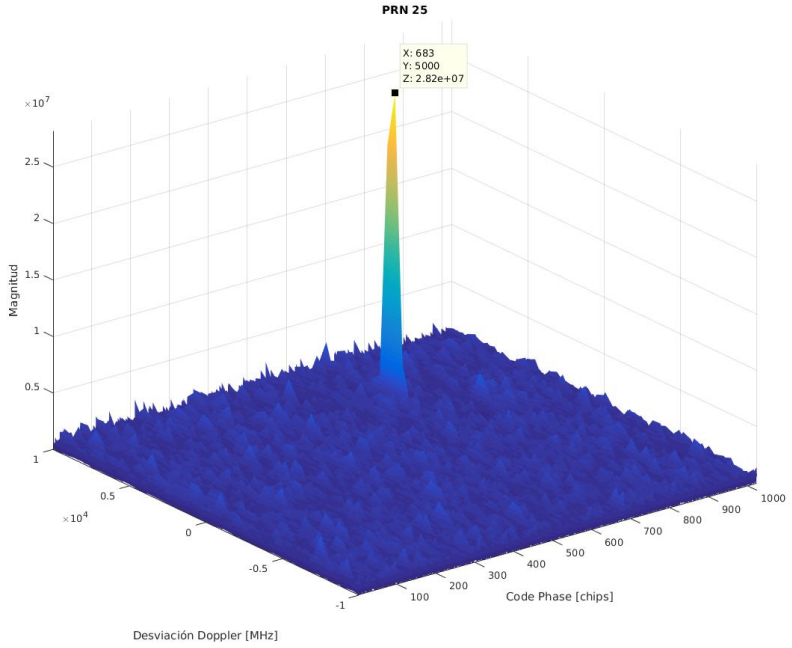

Figura 4.13: Adquisición señal simulada

de dicho receptor.

El receptor implementado por este programa puede funcionar tanto a partir de un archivo que contenga una señal **GPS** como desde una señal capturada a tiempo real. A parte de todo esto, el programa nos permitirá obtener las distintas señales de cada uno de los bloques, entre ellas la señal que capturamos. Podemos observar el esquema del programa en la figura 1.3.

La señal capturada, es la que podemos encontrar a la salida del bloque *sigan conditioner* y será la que procesemos mediante los algoritmos diseñados en Matlab que podemos ver en el Anexo 1. Con respecto a los algoritmos diseñados en las dos anteriores simulaciones, en esta, al tratarse de una señal real con una mayor duración, se aplicará un umbral de decisión que nos permita filtrar las adquisiciones no válidas (las que no superen el umbral definido).

Antes de mostrar los resultados obtenidos con esta señal, vamos a realizar un pequeño inciso en el archivo de configuración que sirve para configurar los distintos bloques del programa (Ver Fig. 4.14). En un principio, los parámetros de configuración de los distintos bloques se han ido explicando a lo largo de este documento.

De todos los parámetros que podemos ver en la figura 4.14 voy a comentar los siguientes:

- SignalSource.implementation. Este parámetro indica el como se implementa el programa, si desde un archivo o desde un **SDR** utilizando el bloque *osmosdr* de **GNURadio**.
- SignalSource.sampling_frequency. Es la frecuencia de muestreo utilizada, cuyo valor es de 2 Mhz debido a que a una mayor frecuencia para la precisión y los rangos de búsqueda del desplazamiento Doppler utilizados, mi ordenador perdía muestras.

```
[GNSS-SDR]

;######### GLOBAL OPTIONS ##################
GNSS-SDR.internal_fs_hz=2000000

;######### SIGNAL_SOURCE CONFIG #############
SignalSource.implementation =        Osmosdr_Signal_Source
SignalSource.osmosdr_args = hackrf,bias = 1

SignalSource.freq =                  1575420000
SignalSource.gain =                  40
SignalSource.rf_gain =               40
SignalSource.if_gain =               30
SignalSource.sampling_frequency =    2000000
SignalSource.AGC_enabled =           false
SignalSource.samples =               0
SignalSource.item_type =             gr_complex
SignalSource.repeat =                false
SignalSource.enable_throttle_control = false
SignalSource.dump =                  true
SignalSource.dump_filename =         ./signal_source.dat

;######### SIGNAL_CONDITIONER CONFIG ############
SignalConditioner.implementation =   Signal_Conditioner

;######### DATA_TYPE_ADAPTER CONFIG ############
DataTypeAdapter.implementation =     Pass_Through

;######### INPUT_FILTER CONFIG ############
InputFilter.implementation =         Freq_Xlating_Fir_Filter
InputFilter.decimation_factor =      1
InputFilter.input_item_type =        gr_complex
InputFilter.output_item_type =       gr_complex
InputFilter.filter_type =            bandpass

InputFilter.taps_item_type =         float
InputFilter.number_of_taps =         5
InputFilter.number of bands =        2
InputFilter.band1_begin =            0.0
InputFilter.band1_end =              0.85
InputFilter.band2_begin =            0.9
InputFilter.band2_end =              1.0
InputFilter.ampl1_begin =            1.0
InputFilter.ampl1_end =              1.0
InputFilter.ampl2_begin =            0.0
InputFilter.ampl2_end =              0.0
InputFilter.band1_error =            1.0
InputFilter.band2_error =            1.0
InputFilter.grid_density =           16

;######### RESAMPLER CONFIG ############
Resampler.implementation=Pass_Through

;######### CHANNELS GLOBAL CONFIG ############
Channels_1C.count =                  8
Channels.in_acquisition =            1
Channel.signal =                     1C

;######### ACQUISITION GLOBAL CONFIG ############
Acquisition_1C.implementation=GPS_L1
_CA_PCPS_Acquisition_Fine_Doppler
Acquisition_1C.item_type =           gr_complex
Acquisition_1C.if =                  0
Acquisition_1C.sampled_ms =          0.5
Acquisition_1C.threshold =           0.015
Acquisition_1C.doppler_max =         15000
Acquisition_1C.doppler_min =         -15000
Acquisition_1C.doppler_step =        500
```

Figura 4.14: Archivo de configuración de los parámetros de captura y adquisición

- SignalSource.item_type. Se trata del formato de las muestras que tendrá la señal capturada. Para este modo de utilización el programa sólo le es válido un tipo de formato, en cambio, para la utilización desde archivo hay varios tipos de formato. El formato *gr_complex* equivale a muestras complejas con parte real e imaginaria de tipo *float* ($[S_0^I + jS_0^Q]$, $[S_1^I + jS_1^Q]$, ...)

- Acquisition_1C.implementation. Se trata del algoritmo utilizado en la adquisición, en este caso se utilizará el **PCPS**, explicado en el apartado *4.1.3*.

- Acquisition_1C.threshold. Es el umbral de adquisición dado, en este programa el umbral se calcula mediante la división del máximo de la matriz de correlación con la potencia de la señal.

- Acquisition_1C.doppler_max y Acquisition_1C.doppler_min. Son los rangos de búsqueda del desplazamiento Doppler. Se dijo que el rango de búsqueda del desplazamiento Doppler para una receptor y transmisor que se están moviendo es $\pm 10Khz$, en este caso el receptor no se mueve pero se ha añadido más frecuencias de búsqueda debido a que el **SDR** introduce una desviación en frecuencia.

- Acquisition_1C.doppler_step. El *step* o la precisión del algoritmo es el valor clásico, aumentar la precisión provocaría que se perdiesen muestras durante la ejecución del programa. Teniendo en cuenta los rangos anteriores y el espaciado, tanto este software, como mi función el algoritmo buscará entre 61 posibles frecuencias portadoras.

Una vez explicados algunos de los parámetros del archivo de configuración, se procede a ejecutar el programa. Durante la ejecución del programa en el terminal de Linux se podrá observar que satélites esta detectando, y en el momento que detecte 4 y obtenga el mensaje de navegación correspondiente para cada uno de ellos, el programa finaliza. Una vez finalizado el programa se obtienen los diferentes archivos que se han especificado en el archivo de configuración, en este caso el archivo a la salida del bloque *signal conditioner* (Ver Fig. 4.14 parámetro *SignalSource.dump*). A parte de estos archivos, también se

obtiene un archivo kml con la posición del receptor. En la figura 4.15 tenemos una muestra del terminal de Linux del día de la captura de la señal **GPS** en donde aparecen los satélites detectados.

```
adrian@Adrian:~$ gnss-sdr --
config_file=/home/adrian/Escritorio/captura.conf
linux; GNU C++ version 5.3.1 20151219; Boost_105800; UHD_
003.009.002-0-unknown

Initializing GNSS-SDR v0.0.9 ... Please wait.
Logging will be done at "/tmp"
Use gnss-sdr --log_dir=/path/to/log to change that.
OsmoSdr arguments: hackrf,bias=1
gr-osmosdr 0.1.4 (0.1.4) gnuradio 3.7.9
built-in source types: file osmosdr fcd rtl rtl_tcp uhd miri
hackrf bladerf rfspace airspy redpitaya
Using HackRF One with firmware 2014.08.1
Enabled antenna bias voltage
Actual RX Rate: 2000000.000000 [SPS]...
Actual RX Freq: 1575420000.000000 [Hz]...
PLL Frequency tune error 0.000000 [Hz]...Actual RX Gain:
14.000000 dB...
Using Volk machine: avx2_64_mmx_orc
Tracking start on channel 0 for satellite GPS PRN 01 (Block IIF)
Tracking start on channel 1 for satellite GPS PRN 05 (Block IIR-
M)
Current input signal time = 1 [s]
Loss of lock in channel 0!
Current input signal time = 2 [s]
Loss of lock in channel 1!
Tracking start on channel 2 for satellite GPS PRN 05 (Block IIR-
M)
Current input signal time = 3 [s]
Current input signal time = 4 [s]
Current input signal time = 5 [s]
Current input signal time = 6 [s]
Current input signal time = 7 [s]
Current input signal time = 8 [s]
Current input signal time = 9 [s]
Tracking start on channel 0 for satellite GPS PRN 30 (Block IIF)
Current input signal time = 10 [s]
Current input signal time = 11 [s]
NAV Message: received subframe 5 from satellite GPS PRN 05
(Block IIR-M)
Current input signal time = 12 [s]
Current input signal time = 13 [s]
Current input signal time = 14 [s]
Current input signal time = 15 [s]
Current input signal time = 16 [s]
Current input signal time = 17 [s]
NAV Message: received subframe 1 from satellite GPS PRN 05
(Block IIR-M)
Current input signal time = 18 [s]
Current input signal time = 19 [s]
Current input signal time = 20 [s]
Current input signal time = 21 [s]
Tracking start on channel 1 for satellite GPS PRN 28 (Block IIR)
Current input signal time = 22 [s]
Current input signal time = 23 [s]
NAV Message: received subframe 2 from satellite GPS PRN 05
(Block IIR-M)
NAV Message: received subframe 2 from satellite GPS PRN 30
(Block IIF)
Current input signal time = 24 [s]
Current input signal time = 25 [s]
Current input signal time = 26 [s]
Current input signal time = 27 [s]
Current input signal time = 28 [s]
Current input signal time = 29 [s]
NAV Message: received subframe 3 from satellite GPS PRN 05
(Block IIR-M)
Current input signal time = 30 [s]
Current input signal time = 31 [s]
Current input signal time = 32 [s]
Current input signal time = 33 [s]
Tracking start on channel 3 for satellite GPS PRN 20 (Block IIR)
Tracking start on channel 4 for satellite GPS PRN 21 (Block IIR)
Current input signal time = 34 [s]
Loss of lock in channel 4!
Loss of lock in channel 3!
Current input signal time = 35 [s]
NAV Message: received subframe 4 from satellite GPS PRN 28
(Block IIR)
NAV Message: received subframe 4 from satellite GPS PRN 05
(Block IIR-M)
Current input signal time = 36 [s]
Current input signal time = 37 [s]
Current input signal time = 38 [s]
Current input signal time = 39 [s]
Current input signal time = 40 [s]
Current input signal time = 41 [s]
NAV Message: received subframe 5 from satellite GPS PRN 05
(Block IIR-M)
NAV Message: received subframe 5 from satellite GPS PRN 30
(Block IIF)
Current input signal time = 42 [s]
Current input signal time = 43 [s]
Current input signal time = 44 [s]
Current input signal time = 45 [s]
Current input signal time = 46 [s]
Current input signal time = 47 [s]
NAV Message: received subframe 1 from satellite GPS PRN 05
(Block IIR-M)
NAV Message: received subframe 1 from satellite GPS PRN 28
(Block IIR)
NAV Message: received subframe 1 from satellite GPS PRN 30
(Block IIF)
Current input signal time = 48 [s]
Current input signal time = 49 [s]
Current input signal time = 50 [s]
```

Figura 4.15: Terminal de Linux donde aparecen los satélites detectados

Como podemos ver en la figura 4.15, durante los 50 segundos de duración, se detectan los satélites 05, 28 y 30 y es entre los segundos 47 y 48 en donde tenemos la detección de los tres, será este el intervalo de tiempo que analizamos con nuestro algoritmo de Matlab. Para comprobar que esta detección es correcta, en paralelo a la detección se ha utilizado una aplicación de móvil que nos muestra que satélites está detectando el móvil para posicionarnos. (Ver Fig. 4.16)

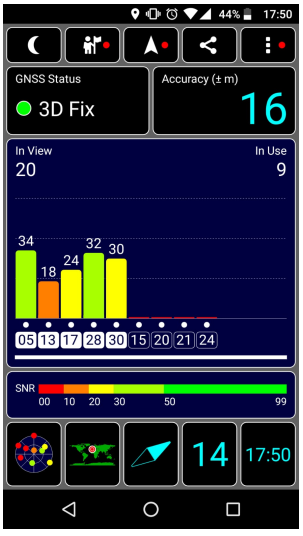

Figura 4.16: Captura de pantalla de la app del móvil para corroborar los satélites detectados

Finalmente, al aplicar el algoritmo para la señal capturada durante los segundos 47 a 48 en tramos de señal de 1 ms, se obtienen $1s/1ms = 1000$ tramos a analizar para 32 códigos distintos, lo que serían 32000 imágenes, esta es la razón por la que el umbral de detección es tan importante en esta simulación.

Para esta simulación se han aplicado 2 métodos distintos a la hora de trabajar con los tramos de 1 ms de la señal. El primero de ellos es escogiendo estos tramos sin ningún tipo de solape de las muestras de la señal capturada. (Ver Fig. 4.17)

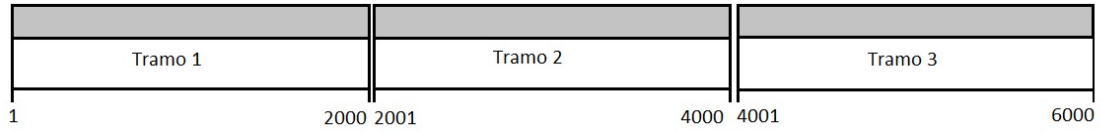

Figura 4.17: Tramos de señal de 1 ms sin solape

El segundo método es utilizar la mitad de las muestras de la señal de un tramo en el siguiente. (Ver Fig. 4.18)

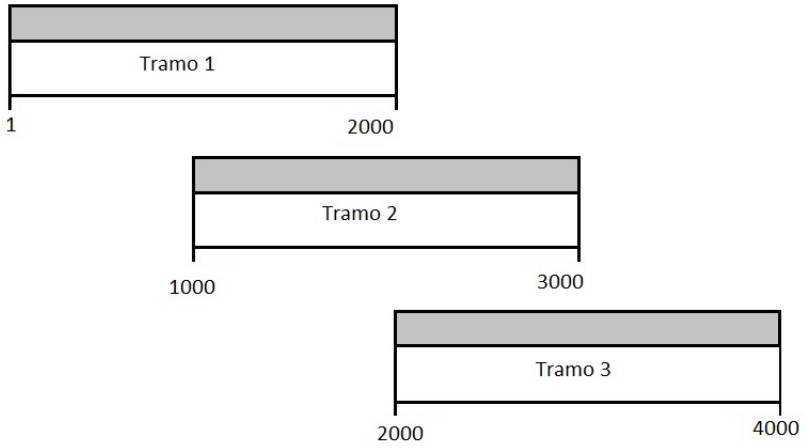

Figura 4.18: Tramos de señal de 1 ms con solape

Aplicando el algoritmo al primer método se observan las gráficas de adquisición del **PRN 5** en el segundo 47.932 (Ver Fig. 4.19), el **PRN 28** en el segundo 47.548 (Ver Fig. 4.20) y el **PRN 30** en el segundo 47.457. (Ver Fig. 4.21)

En cambio, al aplicar el segundo método se observan un mayor número de gráficas con adquisición positiva debido, entre otras cosas, a que en este método estamos realizando más búsquedas ya que en 1 s nos aparecen más tramos. Las gráficas de adquisición obtenidas para los mismos **PRN** serían: el **PRN 5** en el segundo 47.931 (Ver Fig. 4.22), el **PRN 28** en el segundo 47.547 (Ver Fig. 4.23) y el **PRN 30** en el segundo 47.456. (Ver Fig. 4.24)

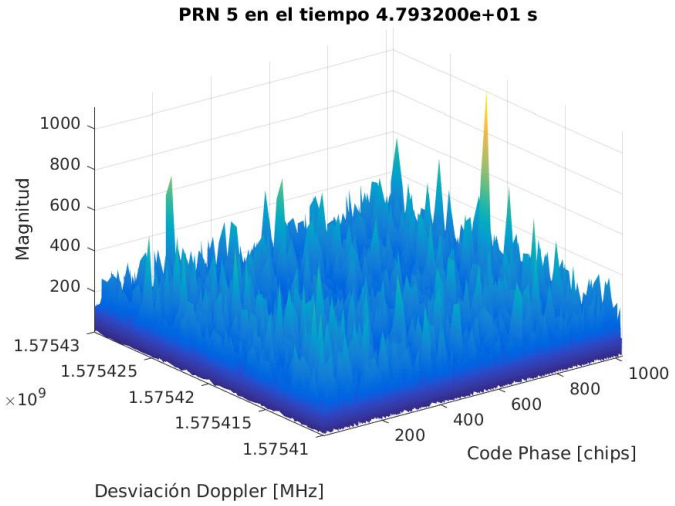

Figura 4.19: Adquisición señal capturada: PRN3 segundo 47.932

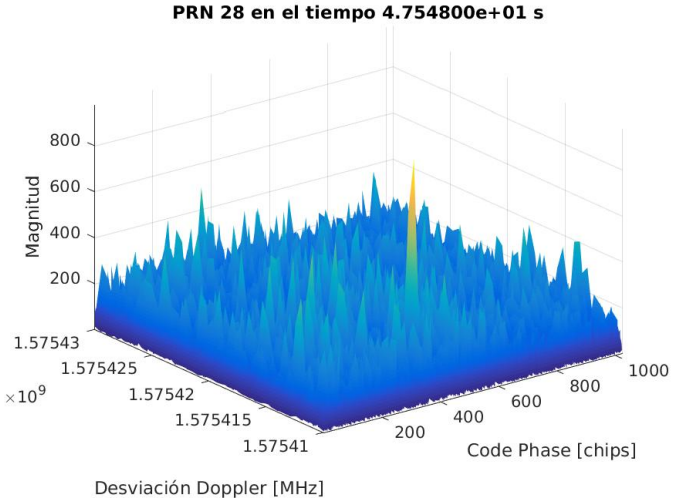

Figura 4.20: Adquisición señal simulada: PRN28 segundo 47.548

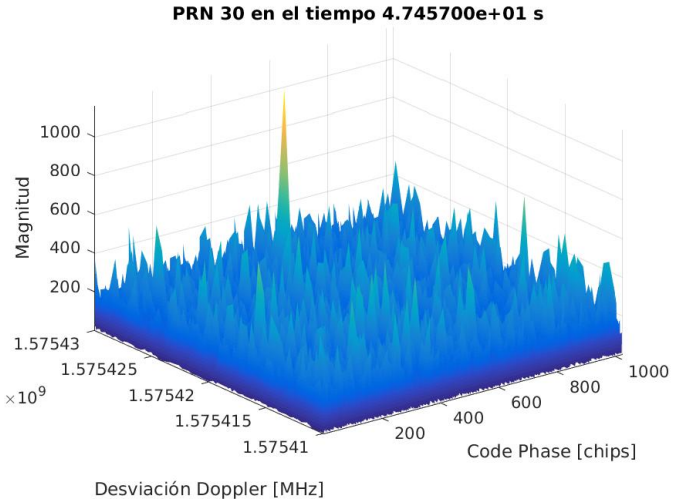

Figura 4.21: Adquisición señal simulada: PRN30 segundo 47.457

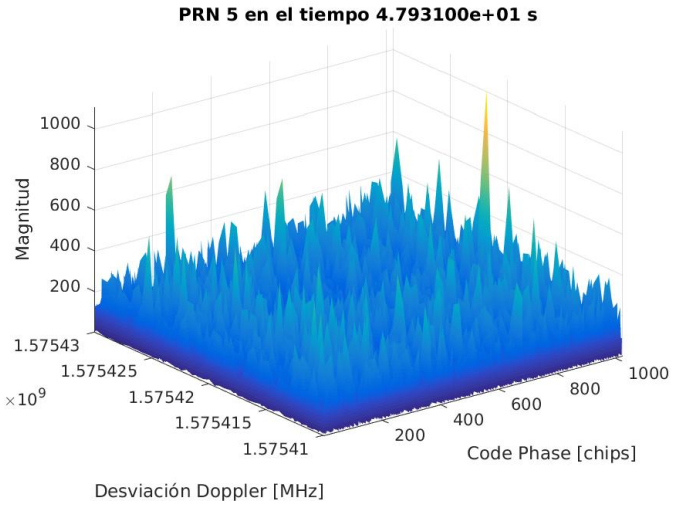

Figura 4.22: Adquisición señal capturada: PRN3 segundo 47.931

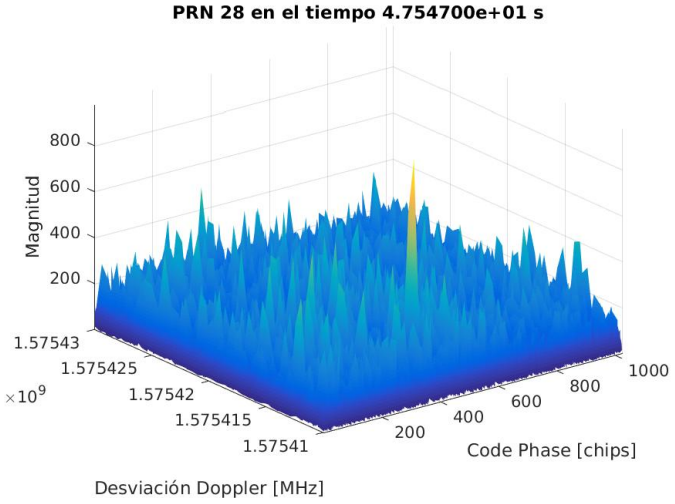

Figura 4.23: Adquisición señal simulada: PRN28 segundo 47.547

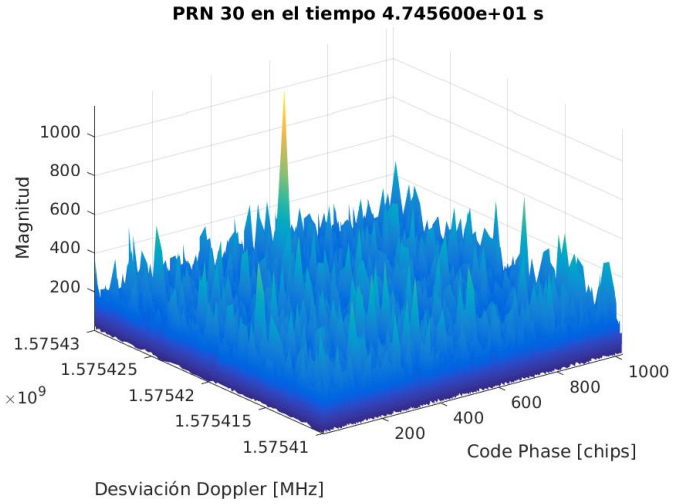

Figura 4.24: Adquisición señal simulada: PRN30 segundo 47.456

Capítulo 5

Conclusiones

Con la realización del presente Trabajo Fin de Grado (TFG) se ha abordado el estudio del sistema **GPS**, así como el funcionamiento de los receptores, con el fin de poder replicar uno de los módulos de dicho receptor, en concreto el módulo de adquisición, sobre distintos tipos de señales **GPS**.

Durante el transcurso del TFG, ha sido necesario aprender más profundamente acerca de los sistemas **GNSS**, en concreto el sistema **GPS**, que lleva funcionando desde hace mucho tiempo y cuyo uso y aplicaciones son utilizadas a diario por todo el mundo. Así como el uso de una gran variedad de *software* diferente para poder realizar distintas simulaciones y comprobaciones.

Gracias a este estudio, se ha conocido en detalle el porque del uso de ciertas técnicas empleadas en este sistema y la forma que tienen los receptores de procesar la señal a partir de dichas técnicas, llegando a implementar el bloque de adquisición de un receptor para, finalmente, obtener los parámetros de desplazamiento Doppler y retraso del código que este bloque proporciona.

Personalmente, considero este primer contacto con las comunicaciones por satélite como un contacto positivo y como un punto de partida para realizar proyecto más específicos en el futuro.

5.1 Trabajos futuros

Con todo lo visto en este documento se podrían realizar una serie de mejoras, tanto a los bloques del receptor que no hemos podido abordar, como con cosas más especificas del propio bloque de adquisición.

Una posible ampliación sería, a partir del bloque de adquisición, realizar en un futuro el diseño del resto de bloques del receptor hasta finalmente poder conseguir situar la posición de dicho receptor.

También se podría trabajar sobre el mismo bloque de adquisición centrándonos de forma más especifica en diferentes aspectos vistos en el TFG como podrían ser: las ventajas e inconvenientes de los distintos algoritmos, centrarse en la elección de los umbrales de detección que aumenten la probabilidad de adquisición y hagan que la probabilidad de falsas alarmas sea lo menor posible, la captura de señales **GPS** a partir de diferentes esquemas de antenas como *arrays*, etc.

Anexos

Anexos A

Anexo I: Código de MATLAB

```
function g = cacode(sv,fs)
%
%function G=CACODE(SV,FS)
%Generates 1023 length C/A Codes for GPS PRNs 1-37
%-------------------------------------------------------------------
%g:  nx1023 matrix- with each PRN in each row with symbols 1 and
%    0
%
%sv: a row or column vector of the SV's to be generated valid
%    entries are 1 to 37
%
%fs: optional number of samples per chip (defaults to 1),
%    fractional samples allowed, must be 1 or greater.
%
%For multiple samples per chip, function is a zero order hold.
%-------------------------------------------------------------------
%For example to generate the C/A codes for PRN 6 and PRN 12 use:
%g=cacode([6 12]);
%
%and to generate the C/A codes for PRN 6 and PRN 12 at 5 MHz use
%g=cacode([6 12],5/1.023)
%
%
%For more information refer to the GPS SPS Signal Specification
%http://www.navcen.uscg.gov/pubs/gps/sigspec/default.htm

if nargin<2
   fs = 1;
end

if (max(sv)>37) || (min(sv)<1) || (min(size(sv))~=1)
   error('sv must be a row or column vector with integers
   between 1 and 37\n')
end

if fs<1
   error('fs must be 1 or greater\n')
end

%Force integers
testint = round(sv)-sv;
if testint ~= 0
   warning('non-integer value entered for sv, rounding to closest
```

```
    integer\n');
    sv = round(sv);
end

% Table of C/A Code Tap Selection (sets delay for G2 generator)
tap = [2  6;          % 1
3  7;                % 2
4  8;                % 3
5  9;                % 4
1  9;                % 5
2  10;               % 6
1  8;                % 7
2  9;                % 8
3  10;               % 9
2  3;                % 10
3  4;                % 11
5  6;                % 12
6  7;                % 13
7  8;                % 14
8  9;                % 15
9  10;               % 16
1  4;                % 17
2  5;                % 18
3  6;                % 19
4  7;                % 20
5  8;                % 21
6  9;                % 22
1  3;                % 23
4  6;                % 24
5  7;                % 25
6  8;                % 26
7  9;                % 27
8  10;               % 28
1  6;                % 29
2  7;                % 30
3  8;                % 31
4  9];               % 32

%% G1 LFSR: x^10+x^3+1
s = [0  0  1  0  0  0  0  0  0  1];
n = length(s);
g1 = ones(1,n); %Initialization vector for G1
L = 2^n-1;

%% G2 LFSR: x^10+x^9+x^8+x^6+x^3+x^2+1
t = [0  1  1  0  0  1  0  1  1  1];
q = ones(1,n);  %Initialization vector for G2

%% Generate C/A Code sequences:
tap_sel = tap(sv,:);
for inc = 1:L
    g2(:,inc) = mod(sum(q(tap_sel),2),2);
    g(:,inc) = mod(g1(n)+g2(:,inc),2);
    g1 = [mod(sum(g1.*s),2) g1(1:n-1)];
    q = [mod(sum(q.*t),2) q(1:n-1)];
end

%% Upsample to desired rate
if fs ~= 1
```

```
    % Fractional upsampling with zero order hold
    index = 0;
    for cnt = 1/fs:1/fs:L
        index = index+1;
        if ceil(cnt) > L    % Traps a floating point error in index
            gfs(:,index) = g(:,L);
        else
            gfs(:,index) = g(:,ceil(cnt));
        end
    end
    g = gfs;
end

end
```

```
function acquisition_plot(ts, fs, t_chip, f, path)
%-------------------------------------------------------------------
% Esta función sirve para hacer el plot de la adquisición de la
% señal GPS.
%
% Entradas:
% ts:       - Periodo de muestreo
% fs:       - Frecuencia de muestreo
% t_chip:   - Tiempo de chip (aprox 1us)
% f:        - Vector de frecuencias con el Doppler
% path:     - Ruta donde quieres guardar las capturas realizadas
%             por el plot
%-------------------------------------------------------------------

figure('visible','off')

x_axis = ceil(((ts)*(1:(fs/1000)))/(t_chip));
y_axis = f;
s = surf(x_axis,y_axis,u);

set(s,'EdgeColor','none','Facecolor','interp');

axis([min(x_axis) max(x_axis) min(y_axis) max(y_axis) min(min(u))
      max(max(u))]);
caxis([0 max(max(u))]);

xlabel('Code Phase [chips]');
ylabel('Desviación Doppler [MHz]');
zlabel('Magnitud');
text = sprintf('PRN %i en el tiempo %d ms',m, k);
title(text);

FileName = sprintf('PRN%i_tiempo_%dms.jpg',m,k);
baseFileName = [path FileName];
saveas(s,baseFileName);

end
```

```
function [signal] = read_file( archivo, total_size )
%-------------------------------------------------------------------
% Esta función sirve para leer los archivos .dat pertenecientes
```

```
% del programa GNSS-SDR al capturar las señales a tiempo real.
% El tipo de datos del archivo .dat son gr_complex o float 16
% bits conparte real e imaginaria.
%
% Entradas:
% archivo:        - Nombre del archivo
% total_size:     - Núm de muestras totlaes que queremos de
%                   nuestro archivo
%
% Salidas:
% signal:         - Señal equivalente a la que llegará al receptor
%-------------------------------------------------------------------

fid = fopen(archivo,'rb');
x = fread(fid,[2, total_size],'float32');
fclose(fid);

signal = x(1,:) + x(2,:)*1i;

end
```

```
%% Adquisición
% Primer script de la adquisición de una señal ejemplo muestreada
% a 11.999MHz y a la frecuencia intermedia de 3.563MHz.
%
% Esta señal esta preparada para trabajarse en Matlab (.mat) y,
% entre otros PRN, se encuentran formada por el 1 y el 10.

%% Datos básicos
load data.mat                       % Cargar señal .mat
fs = 11.999e6;                      % Frecuencia de muestreo
f_if = 3.563e6;                     % Frecuencia intermedia
fmin_Doppler = -10e3;               % Rango de f Doppler máx a buscar
fmax_Doppler = 10e3;                % Rango de f Doppler mín a buscar
step = 500;                         % Paso para la búsqueda de la
                                    % desviación Doppler
%% Inicialización
signal = double(data');

Nchips = 1023;                      %Núm de chips en 1 código C/A

%% Frecuencias:
f_chip = 1.023e6;                   % Frecuencia de chip (gold rate)
% Frecuencia Doppler a compensar en la señal
f = (f_if+fmin_Doppler):step:(f_if+fmax_Doppler);

%% Tiempos:
ts = 1/fs;                          % Tiempo de muestreo
t_chip = 1/f_chip;                  % Tiempo de chip (aprox 1us)(s/chip)
time = (0:Long_1ms_CA-1)*2*pi*ts;   % Vector de tiempos

%% Muestras
m_Chip = round(fs*t_chip);              %Núm de muestras por chip
Long_1ms_CA = round(Nchips*fs*t_chip); % Long 1 código C/A=1023 chips

%% Matriz de autocorrelación
u = zeros(length(f),Long_1ms_CA); % Inicializar matriz de autocorr

```

```
%%Códigos C/A
CA = cacode((1:32),fs/1023e3);      %Creo todos los PRN de todos los SV
CA(CA==0) = -1;                     %Sustituye los 0 de la matriz por -1

for m = 1:32                        %Todos los PRN de los posibles SV
    %------ DFT C/A Code
    CA_freq = conj(fft(CA(m,:)));   %Complejo conjugado de la
                                    %FFT del primer PRN

    for i = 1:((fmax_Doppler+abs(fmin_Doppler))/step)+1 %2*(10k/500)+1
        %------ Creo el seno y coseno local
        sinCarr = sin(f(i)*time);
        cosCarr = cos(f(i)*time);
        %------ Quitamos la portadora de la señal GPS
        I = sinCarr.*signal;
        Q = cosCarr.*signal;
        %------ Convierte la señal al domino de la frecuencia
        IQ_freq = fft(I+1j*Q);
        %------ Multiplicación de la señal I/Q con el PRN local
        IQ_CA_freq = IQ_freq.*CA_freq;
        %------ IFFT para obtener la autocorrelación
        u(i,:) = abs(ifft(IQ_CA_freq)).^2;
     end

     [valor posicion] = max(u(:));      %Pos del máx de la matriz
     [x y] = ind2sub(size(u),posicion); %Obtiene pos en x e y del máx

     frequency(m) = f(x)                %Aprox de la f de la señal

     codephase = y-1;          %Restraso código respecto a 11999 muestras
     codePhase1023(m) = round((codephase/fs)*1.023e6)  %Restraso del
                                                       %código C/A
     acquisition_plot(ts, fs, t_chip, f)
end
```

```
function [Incoming] = gen_signal(fs, bits, sat, vel, tipo_Doppler, var,
                                 codephaseSAT, fase)
%-------------------------------------------------------------------------
%Esta función sirve para generar/simular "x" bits de una señal GPS.
%
%Entradas:
%fs             - Frecuencia de muestreo [muestras/s]
%bits:          - Bits de la señal que queremos crear
%sat            - Número del satélite del que queremos su PRN
%vel            - Velocidad del satélite (para f Doppler) [m/s]
%tipo_Doppler   - 1 = Doopler cte y 0 = Doppler variable
%var            - Varianza del ruido
%codephaseSAT   - Retraso de código impuesto para la simulación
%fase           - Fase de la señal que llega al RX
%
%Salida
%Incoming:      - Señal simulada bandabase
%-------------------------------------------------------------------------
%%------------------------------ Datos básicos:
Nchips = 1023;                  %Núm de chips en 1 código C/A

%Frecuencias:
fL1 = 154*10.23e6;              %Frecuencia portadora L1
```

```
f_chip = 1.023e6;              % Frecuencia de chip (gold rate)

% Tiempos:
ts = 1/fs;                     % Tiempo de muestreo
t_chip = 1/f_chip;             % Tiempo de chip (aprox 1us)(s/chip)

% Muestras
Long_1ms_CA = Nchips*fs*t_chip;    % Long 1 código C/A = 1023 chips C/A
Long_20ms_CA = 20*Long_1ms_CA;     % Long de 20 códigos C/A = 1 bit

%%------------------------------ Generación de la Frecuencia Doppler:
c = 299792458;                 % Velocidad exacta de la luz en [m/s]
doppler = vel*fL1/c;           % Doppler [Hz] - (vel sat/c)*f portadora

if tipo_Doppler == 1
% Opción 1: Frecuencia Doppler cte en todas las muestras
   fdop(1:bits*Long_20ms_CA) = doppler;    % Crea un vector de frec
                                           % Doppler cte
else
      if tipo_Doppler == 0
% Opción 2: Frecuencia Doppler variable
      % Primera mitad tiene Doppler > 0
      fdop(1:bits*Long_20ms_CA/2) = doppler;
      % Segunda mitad tienen Doppler < 0
      fdop((bits*Long_20ms_CA/2)+1:bits*Long_20ms_CA) = -doppler;
      end
end

%%------------------------------ Generación de la señal GPS en el satélite
%% Códiog C/A
CA = cacode(sat,fs/1023e3);          % Código C/A
CA(CA==0) = -1;                      % Cambio los 0 por -1s

CA_bits = repmat(CA,[1,bits*20]);  % Repito un periodo de código C/A 1ms
                                   % tanto como bits quiera simular
%% Datos de navegaión
for i = 0:bits-1
% Generación de 1 bit del msj de navegación (puede ser un 1 o un 0)
x = randi([0,1]);              % Bit del msj de navegación aleatorio 0 y 1
x(x==0) = -1;                  % Cambiar los 0 por -1

% Ampliación de las muestras msj de navegación a su equivalente de 20ms
nav_1_bit(1:Long_20ms_CA) = x;

% Concatenar los "x" bits de la señal
nav(Long_20ms_CA*i+1:Long_20ms_CA*(i+1)) = nav_1_bit;
end

%% Creación señal a la salida del sat:
% Señal a la salida del sat:
signal_tx = CA_bits.*nav;          % Señal salida del sat banda base

%%------------------------------ Efectos producidos por el canal
%% Ruido
ruido = sqrt(var/2).*randn(1,bits*Long_20ms_CA)+1j*sqrt(var/2).
        *randn(1,bits*Long_20ms_CA);

signal_rx = signal_tx + ruido;     % Señal a la entarda del rx con ruido

```

```
%% Desfase del código C/A
signal_rx = circshift(signal_rx,[0 codephaseSAT]);

%%------------------------------ Espectro de la señal GPS que llega al RX
fourier = fftshift(fft(signal_rx));
modulo = abs(fourier);
fi = (0:bits*Long_20ms_CA-1)-bits*Long_20ms_CA/2;

plot(fi,modulo);          % Espectro de la señal GPS
figure

%%------------------------------ Filtro paso bajo
w = (2e6)/(fs/2);                          % Freq corte del filtro
[B,A] = cheby2(3,20,w,'low');              % LPF Chebyshev de tipo 2

signal_rx_filtrada = filter(B,A,signal_rx); % Señal GPS filtrada

%%------------------------------ Espectro de la señal GPS filtrada
fourier=  fftshift(fft(signal_rx_filtrada));
modulo = abs(fourier);

plot(fi,modulo);          % Espectro de la señal GPS filtrada

%%------------------------------ Señal que entra al bloque de adquisición
% Libro: A Software Defined Receiver - pág. 78 (muestras totales)
n = 0:bits*Long_20ms_CA-1;

% Incomming = fórmula de GNSS-SDR del bloque adquisición
Incoming = signal_rx_filtrada.*exp(1j*(2*pi*(ts*fdop.*n)+fase));

end
```

```
function [frequency, codePhase1023] = adquisicion_signal(fs, f_if,
                          fmin_Doppler, fmax_Doppler, step, Incoming)
%-------------------------------------------------------------------------
% Adquisicion de una señal GPS apartir de una señal simulada o no
%
% Entrada:
% fs:               - Frecuencia de muestreo
% f_if:             - Frecuencia central sin Doppler
% fmin_Doppler:     - Frecuencia mí Doppler a buscar
% fmax_Doppler:     - Frecuencia máx Doppler a buscar
% step:             - Separación para el barrido de la frecuencia
% Incoming          - Señal a la entrada del bloque de adquisición
%
% Salida:
% frequency:        - Valor de la f de la señal que llega al receptor
% codePhase:        - Valor del retraso del código C/A
%-------------------------------------------------------------------------

%% Datos básicos:
Nchips = 1023;                %Núm de chips en 1 código C/A

% Frecuencias:
f_chip = 1.023e6;             % Frecuencia de chip (gold rate)
% Frecuencia Doppler a compensar en la señal
f = (f_if+fmin_Doppler):step:(f_if+fmax_Doppler);

```

```
% Tiempos:
ts = 1/fs;                      % Periodo de muestreo
t_chip = 1/f_chip;              % Periodo de chip (aprox 1us)(s/chip)

% Muestras
Long_1ms_CA = round(Nchips*fs*t_chip);   % Long 1 código C/A=1023 chips

% Vector de tiempos
time = (0:(2*Long_1ms_CA-1))*2*pi*ts;

%% Leer la señal
signal = Incoming(1:2*Long_1ms_CA);      % Leemos 2ms de la señal

%% Códigos C/A
%------- Genera todos los códgios C/A de todos los SV
CA = cacode((1:32),fs/1023e3);     % Creo todos los PRN los SV
CA(CA==0) = -1;                    % Sustituye los 0 de la matriz por -1

for m = 1:32                       % Todos los PRN de los posibles SV
    for i = 1:((fmax_Doppler+abs(fmin_Doppler))/step)+1
        %------- Creamos la portadora local
        carrier = signal.*exp(-1j*f(i)*time);
        %------- Convolucionamos la señal (2ms) con 1ms de código C/A
        u(i,:) = conv(carrier,fliplr(CA(m,:)));
    end

    u = u(:,Long_1ms_CA:2*Long_1ms_CA); % Muestras centrales
    u = abs(u).^2;                      % El valor absoluto al cuadrado

    [valor posicion] = max(u(:));       % Pos valor máximo de la matriz
    [x y] = ind2sub(size(u),posicion);  % Posición en x e y del máx

    frequency(m) = f(x);                % Aprox de la f de la señal

    codePhase(m) = y-1;                 % Restraso del código respecto
                                        % a las muestras de código
    codePhase1023(m) = round((codephase/fs)*1.023e6)   % Restraso del
                                                       % código C/A
    acquisition_plot(ts, fs, t_chip, f)

    clear u;
end

end
```

```
function [f_Doppler, codephase_CA] = adquisicion(archivo, fs, f_if,
          fmax_Doppler, fmin_Doppler, step, t_ini, t_fin, umbral, path)
%-------------------------------------------------------------------
% Adquisición de la señal capturada a tiempo real.
%
% Entrada:
% archivo:          - Nombre del archivo a analizar
% fs:               - Frecuencia de muestreo
% f_if:             - Frecuencia intermedia
% fmax_Doppler:     - Rango de f máximo de Doppler
% fmin_Doppler:     - Rango de f mínimo de Doppler
% step:             - espaciado para buscar el desviación Doppler
% t_ini:            - Tiempo inicial de la señal que queremos analizar
```

```
14 % t_fin:            - Tiempo final de la señal que queremos analizar
15 % umbral:           - Valor del umbral de adquisición
16 % path:             - Dirección del directeorio donde se guardarán
17 %                     los plots
18 %
19 % Salidas:
20 % f_Doppler:        - Frecuencia de la señal con el Doppler
21 % codephase_CA:     - Retraso del código C/A
22 %-------------------------------------------------------------------
23 %% Inicialización
24 Nchips = 1023;                %Núm de chips en 1 código C/A
25
26 % Frecuencias:
27 f_chip = 1.023e6;             % Frecuencia de chip (gold rate)
28 % Frecuencia Doppler a compensar en la señal
29 f = (f_if+fmin_Doppler):step:(f_if+fmax_Doppler);
30
31 % Tiempos:
32 ts = 1/fs;                    % Periodo de muestreo
33 t_chip = 1/f_chip;            % Periodo de chip (aprox 1us)(s/chip)
34 t_CA = 1e-3;                  % Tiempo de 1 C/A
35
36 % Muestras
37 m_Chip = round(fs*t_chip);                 %Núm de muestras por chip
38 Long_1ms_CA = round(Nchips*fs*t_chip);     % Long 1 código C/A=1023 chips
39
40 m_ini = t_ini*fs;                          % Muestra de inicio de la señal
41 m_fin = t_fin*fs;                          % Muestra fianl de la señal
42
43 %% Vector de tiempo:
44 time = (0:(2*Long_1ms_CA-1))*2*pi*ts;
45
46 %% Leer señal:
47 total_size = t_fin*fs;                     % Tamaño total del archivo
48 signal = read_file(archivo,total_size+Long_1ms_CA);
49
50 %% Codigos C/A:
51 CA = cacode((1:32),fs/1023e3);      % Creo todos los PRN de todos los SV
52 CA(CA==0) = -1;                     % Sustituye los 0 de la matriz por -1
53
54 %% Procesado
55 for k = m_ini:Long_1ms_CA:m_fin-1
56     % 1ms de la señal GPS corresponde a las muestras de tiempo escogido
57     signal_buffer = signal(k:k+Long_1ms_CA);
58
59
60     for m = 1:32                    % Todos los PRN de los posibles SV
61     %------- DFT C/A Code
62         % Complejo conjugado de la FFT del 1er PRN
63         CA_freq = conj(fft(CA(m,:)));
64
65         for i = 1:((fmax_Doppler+abs(fmin_Doppler))/step)+1
66             %------- Creo el seno y coseno local
67             sinCarr = sin(f(i)*time);
68             cosCarr = cos(f(i)*time);
69             %------- Quitamos la portadora de la señal GPS
70             I = sinCarr.*signal_buffer;
71             Q = cosCarr.*signal_buffer;
72             %------- Convierte la señal al domino de la frecuencia
```

```
73                  IQ_freq = fft(I+1j*Q);
74                  %------ Multiplicación de la señal I/Q con el PRN local
75                  IQ_CA_freq = IQ_freq.*CA_freq;
76                  %------ IFFT para obtener la autocorrelación
77                  u(i,:) = abs(ifft(IQ_CA_freq)).^2;
78              end
79
80              if max(u(:))/mean(u(:)) > umbral
81
82              [valor posicion] = max(u(:));          %Pos del máx de la matriz
83              [x y] = ind2sub(size(u),posicion);     %Valor pos en x e y
84
85              f_Doppler = f(x);                  %Valor del Doppler de la señal
86
87              codephase = y-1;
88              codephase_CA = (codephase/fs)*1.023e6;   %Retraso código 1023
89
90              acquisition_plot(ts, fs, t_chip, f, path)
91              end
92          end
93      end
94
95      end
```

Bibliografía

[1] Blog-gps. `http://www.explorelabs.com/blog/designing-a-gps-receiver/`.

[2] Gps-sdr-sim. `https://github.com/osqzss/gps-sdr-sim`.

[3] Hackrf. `https://greatscottgadgets.com/hackrf/`.

[4] Prn. `https://natronics.github.io/blag/2014/gps-prn/`.

[5] A Annex. Global positioning system standard positioning service signal specification, 1995.

[6] Kai Borre, Dennis M Akos, Nicolaj Bertelsen, Peter Rinder, and Søren Holdt Jensen. *A software-defined GPS and Galileo receiver: a single-frequency approach*. Springer Science & Business Media, 2007.

[7] Dan Boschen. C/a. `https://www.mathworks.com/matlabcentral/fileexchange/14670-gps-c-a-code-generator`.

[8] G Arul Elango, GF Sudha, and Bastin Francis. Weak signal acquisition enhancement in software gps receivers–pre-filtering combined post-correlation detection approach. *Applied Computing and Informatics*, 13(1):66–78, 2017.

[9] Carles Fernández-Prades. Gnss-sdr. `https://gnss-sdr.org/`.

[10] Elliott Kaplan and Christopher Hegarty. *Understanding GPS: principles and applications*. Artech house, 2005.

[11] Jérôme Leclere, Cyril Botteron, and Pierre-André Farine. Modified parallel code-phase search for acquisition in presence of sign transition. In *Localization and GNSS (ICL-GNSS), 2013 International Conference on*, pages 1–6. IEEE, 2013.

[12] José Raúl Machado-Fernández. Software defined radio: Basic principles and applications. *Facultad de Ingeniería*, 24(38):79–96, 2015.

[13] Nicolas Martin, Valéry Leblond, Gilles Guillotel, and Vincent Heiries. Boc (x, y) signal acquisition techniques and performances. In *Proceedings of the 16th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS/GNSS 2003)*, pages 188–198, 2001.

[14] Damian Miralles and Pau Closas. Faster gnss via the sparse fourier transform. 2014.

[15] Pangan Ting, Hui-Ming Wang, and Nan-Sheng Huang. Software defined radio (sdr) architecture for wireless digital communication systems, December 19 2006. US Patent 7,151,925.

[16] Gonçalo Tomé. Post-processed acquisition & tracking of gps c/a l1 signals. 2015.

[17] James Bao-Yen Tsui. *Fundamentals of global positioning system receivers: a software approach*, volume 173. John Wiley & Sons, 2005.